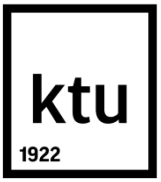


KAUNO TECHNOLOGIJOS UNIVERSITETAS

EKONOMIKOS IR VERSLO FAKULTETAS

Marius Rumčikas

IT PROJEKTŲ LANKSTUMO IR EFEKTYVUMO DERINIMAS TAIKANT AGILE METODOLOGIJĄ

MAGISTRO DARBAS

Darbo vadovas Doc. dr. Mantas Vilkas

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS
EKONOMIKOS IR VERSLO FAKULTETAS

IT PROJEKTŲ LANKSTUMO IR EFEKTYVUMO DERINIMAS TAIKANT AGILE METODOLOGIJĄ

Technologijų vadyba (621N20032)

MAGISTRO DARBAS

Darbą atliko

VMV-4 gr., Marius Rumčikas

_____

2016 m. .....**d.**

Vadovas

Doc. dr. Mantas Vilkas

_____

2016 m. .....**d.**

Recenzentas

Lekt. dr. Rasa Lalienė

_____

2016 m. .....**d.**

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Ekonomikos ir verslo fakultetas

Marius Rumčikas

Technologijų vadyba (621N20032)

Baigiamojo magistro darbo „IT projektų lankstumo ir efektyvumo derinimas taikant Agile metodologiją“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ____ m. _____ ____ d.
Kaunas

Patvirtinu, kad mano **Mariaus Rumčiko** baigiamasis magistro darbas tema „IT projektų lankstumo ir efektyvumo derinimas taikant Agile metodologiją“ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____ _____
(vardą ir pavardę įrašyti ranka) *(parašas)*

Rumčikas, Marius. Combining Flexibility and Efficiency of IT Projects Using Agile Methodology. Master's Final Thesis in Technology Management / supervisor Doc. Dr Mantas Vilkas. Department management, the School of Economics and Business, Kaunas University of Technology.
Social Sciences: Management and Administration
Key words: *Agile, methodology, IT, efficiency, flexibility* .
Kaunas, 2016. 56 p.

SUMMARY

Agile is an Information Technology (IT) project management methodology which promotes flexibility and efficiency. The people who support the Agile methodology think that strict rules and difficult processes make the management of a project hard. According to them, projects should make the process more efficient and should make a product's quality better. This can't be achieved if development process is strictly structured and regulated. To that end, Agile stimulates project participant's cooperation and does not project process formalization.

Research aim. To investigate and evaluate Agile methods impact making IT projects flexible and efficient and to make recommendations.

Research tasks.

1. To analyze the problems of the Agile methods.
2. To analyze the scientific literature in order to reveal the essence of the Agile methods and application characteristics.
3. Investigate coherence between efficiency and flexibility for project realization using Agile methods.

Agile iterations ideology provides the flexibility and efficiency of harmonization. In every iteration the purpose of the task is defined and it is very important that the goal of this task be reachable and all members of the team try to reach that goal. When the aim of the task is known, team members distributes roles and tasks. The goal aim motivates, organizes and disciplines the team which ensures efficiency of each iteration. Fixed iteration time teaches programmers to assess how much time they need to make task done. When team have knowledge of how much time each need to make some tasks they can distribute tasks and roles more easily and make development more efficient.

In conclusion the studies led to form Agile methodology coherence model of flexibility and efficiency, and confirmed that methodology flexibility promotes team results and efficiency. However it is necessary to consider that you have to involve customer/consumer in to team work in order to achieve that efficiency. Also using more innovative instruments like lean methodology or Kanban method helps to improve efficiency. For future researches, it would be appropriate to conduct research about customer participation and influence for software development quality.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Tyrimo aktualumas. Šiuolaikiniame pasaulyje pasisekimo šansų turi tik tos kompanijos, kurios gali dinamiškai besivystančios rinkos ekonomikos sąlygomis optimizuoti savo verslo procesus. Greitai besivystančių informacinių technologijų amžiuje tam tikslui daugelis organizacijų kuria IT-sistemas užsibrėžtiems tikslams pasiekti. Šiuo metu labai aštrios verslo procesų automatizavimo ir informacinių išteklių valdymo problemos. Dažnai programinių priemonių kūrimo projektai baigiasi viršijant numatytus terminus ir biudžetą, o rezultate sukurtas produktas nepilnai atitinka užsakovo reikalavimus. Dažnai IT-sistema ar jos atskiri moduliai kuriami taip, kad užsakovai pradinėje stadijoje neaiškiai įsivaizduoja galutinius rezultatus ir projektavimo procese formuluoja naujus reikalavimus. Kūrėjai savo ruožtu, priversti peržiūrėti savo sprendimus ir siūlyti realizuoti naujus. Projekte naudojamą infrastruktūrą ir idėjų realizavimo instrumentus taip pat gali tekti atnaujinti. Esminės problemos taip pat yra susijusios su kompanijos darbuotojų, dalyvaujančių uždavinių formulavime, dizaino, paties programinio produkto kūrime, funkcionalumo testavime, jo patikrinime, tarpusavio sąveika. Tokiomis sąlygomis negalima neįvertinti efektyvaus projektų ir pokyčių juose valdymo svarbą. Projektų vadovai, konsultantai, verslo analitikai ir kiti specialistai, nuo kurių priklauso automatizuotos verslo procesų valdymo sistemos sukūrimo sėkmė turi žinoti ir gebėti taikyti šiuolaikines komandinio darbo organizavimo IT-sferoje technologijas ir turėti atitinkamas kompetencijas.

Daugeliu atvejų IT-projekto kūrimas – sudėtingas procesas, reikalaujantis iš specialistų grupės suderinto darbo ir kūrybiškumo. Sudėtingumas – vienas iš svarbiausių klausimų, iškylančių kuriant šiuolaikines programines priemones. Jo sprendimui skirtos nemenkos pastangos informacinių sistemų kūrimo, valdymo ir funkcionavimo kokybės kontrolės priemonių kūrimui. Šiuolaikinės projektų valdymo metodologijos skirtos užtikrinti programinių priemonių kūrimo ir realizavimo procesų rezultatyvumą, racionalumą ir valdymo efektyvumą.

Agile IT projektų valdymo metodologija pagrįsta lankstumu ir efektyvumu. Agile metodologijos šalininkams valdymas yra infrastruktūra, pridedanti nereikalingus darbus ir procesų griežtumą. Metodologijos šalininkų nuomone, projektai turi išnaudoti bet kokią galimybę, kad padidinti produkto kokybę ir jo kūrimo efektyvumą. Tai negali būti pasiekta jei kūrimo procesas yra griežtai struktūrizuotas ir reguliuojamas, todėl tokiu būdu Agile skatina projekto dalyvių bendradarbiavimą, o ne griežtą projekto procesų formalizavimą. Taip pat reikia pažymėti, kad gilios analizės reikalaujantys ir rizikingi projektai gali pareikalauti griežtesnio planavimo, nei agile metodas gali suteikti.

Tyrimo naujumas. Paskutiniame dešimtmetyje buvo pateikta visa eilė agile metodų, tokių, kaip eXtreme Programming (ekstremalus programavimas) (Sfetsos, Angelis, Stamelos, 2006), SCRUM (Vlaanderen, 2011), dinaminė sistemų kūrimo metodologija (DSDM) (Livermore, 2008). Nors šie metodai ir skiriasi savo ypatumais, tačiau jie turi bendrą tikslą leisti projekto komandoms greičiau reaguoti į pokyčius. Kadangi pokyčius pritaikyti projekte po to, kai jo realizavimas pasistūmėjęs į priekį, yra pakankamai brangu, galimybė greitai reaguoti į pokyčius sumažina projekto riziką ir jo kaštus (Lucia, Qusef, 2010). Tuo metu kai Agile metodai yra efektyvūs ir lankstūs tam tikruose kontekstuose, didelių ir sudėtingų programinių produktų kūrimas dažnai reikalauja sisteminės projekto realizavimo disciplinos (griežtumo), kad užtikrinti sėkmę. Iššūkiu vadovams tampa nustatyti ar Agile metodas tinka šiam projekto realizavimui.

Tyrimo problema. Kaip Agile metodai leidžia pasiekti IT projektų lankstumą ir efektyvumą?

Tyrimo tikslas. Ištirti ir įvertinti Agile metodų įtaką IT projektų realizavimo lankstumui ir efektyvumui ir pateikti rekomendacijas Agile metodų panaudojimui.

Tyrimo uždaviniai:

1. Išanalizuoti Agile metodų taikymo problematiką.
2. Išanalizuoti mokslinę literatūrą, siekiant atskleisti Agile metodų esmę ir taikymo ypatumus.
3. Ištirti Agile metodų sąsajas su IT projektų realizavimo efektyvumu ir lankstumu.

Tyrimo metodai:

- mokslinės literatūros analizės;
- pusiau struktūruoto interviu;
- pusiau struktūruoto interviu rezultatų turinio analizės;

Tyrimo rezultatai, jų teorinis ir praktinis reikšmingumas. Šiame darbe buvo pasiūlytas modelis, atskleidžiantis, kaip Agile metodai prisideda prie IT projektų lankstumo ir efektyvumo. Lankstumą užtikrina – komandos narių kompetencijos, vidinė komunikacija, saviorganizacija, komandos narių įsitraukimas, lyderystė, vartotojo dalyvavimas ir inovatyvių instrumentų panaudojimas. Efektyvumą – pateikto galutinio produkto aukštesnė kokybė, trumpesnis produkto pateikimo laikas ir didesnė vertė vartotojui; lankstumą ir efektyvumą užtikrina nuostolių mažinimas.

Darbo struktūra. Šį darbą sudaro įvadas, keturios dalys, išvados ir rekomendacijos Pirmoje dalyje atskleidžiama tradicinių programinės įrangos kūrimui naudojamų metodų

panaudojimo problemos, kurias sąlygoja greitai besikeičiantys reikalavimai kuriamai programinei įrangai. Antroje dalyje pateikiami teoriniai Agile projektų valdymo metodų panaudojimo kuriant programines priemones aspektai. Atskleidžiama tokių metodų esmė ir ypatumai, pateikiami plačiausiai naudojamų metodų aptašymai. Trečioje dalyje pateikiama Agile projektų valdymo metodų taikymo programinės įrangos kūrimui lankstumo ir efektyvumo užtikrinimo tyrimo metodologija. Ketvirtame skyriuje pateikiami tyrimo rezultatų analizė bei šios analizės rezultatų pagrindu sudarytas Agile metodų taikymo IT projektų realizavimui lankstumo ir efektyvumo užtikrinimo modelis.

1. TRADICINIŲ PROGRAMINĖS ĮRANGOS METODŲ NAUDOJIMO PROBLEMATIKA

1.1. Tradiciniai programinės įrangos kūrimo metodai

Kaip teigia D. Pilone ir R. Miles (2007), nepriklausomai nuo to kaip atliekamas programinės įrangos kūrimas, koks būdas naudojamas, visada yra vykdomas problemos sprendimas. Bendru atveju būdas, kuriuos programinės įrangos kūrėjai sprendžia problemas yra tas pats, nepriklausomai nuo problemos ar naudojamo metodo. Problemos sprendimas apima keturias pagrindines veiklas:

- detalių apie problemą surinkimas ir dokumentavimas;
- analizė – problema suvokiama pakankamai detaliai, kad būtų užtikrintas teisingas sprendimas;
- kūrimas – optimalaus problemos sprendimo suradimas ir specifikavimas;
- realizavimas (jei reikalinga) – sprendimo realizavimas atitinkamoje formoje.

Pagrindinė problema programinės įrangos kūrime yra kaip realizuoti tam tikrą informacijos apdorojimo sistemą, panaudojant atitinkamas technologijas ir įvertinant tam tikrus apribojimus. P. Robillard (2009) teigimu, programinės įrangos kūrėjai ar vadovai ne visada supranta (bent jau dalis jų), kad programinės įrangos kūrėjų uždavinys yra spręsti srities problemas. Nors yra probleminės srities supratimo problemos, jos bendru atveju nėra susijusios su programinės įrangos kūrimu. Galima būtų ginčytis, kad nepriklausomai kokia paradigma ar būdas pasirinktas programinei įrangai kurti, tam tikru laipsniu turi būti panaudota kiekviena problemos sprendimo veikla. Iš esmės, kiekvienas programuotojas pereina reikalavimų, analizės, kūrimo ir realizavimo ciklą per ilgą laiką, savaitę, dieną, kelias valandas ar minutes, jis dokumentuoja rezultatus, diskutuoja apie juos su kitais arba tiesiog analizuoja juos pats. Tad šių veiklų išvengti neįmanoma.

Programinės įrangos kūrimo ciklas suteikia aukšto lygio perspektyvą apie tai, kaip skirtingos problemos sprendimo veiklos gali būti apjungtos į fazes, esant individualiam ar komandiniam programinės įrangos kūrimui. D. Unger ir S. Eppinger (2011) pateikia pakankamai išsamų programinės įrangos kūrimo gyvavimo ciklų sąrašą. Kaip, pačius populiariausius jie išskiria: vandens krioklio, iteracinį, augantį iteracinį, evoliucinių prototipų programinės įrangos kūrimo gyvavimo ciklus.

N. Ruparelia (2010) teigimu, vandens krioklio metodas apima nuoseklų etapų atlikimą pilna apimtimi ir perėjimą į kitą fazę. Šis metodas niekuomet nebuvo siūlomas praktiškai naudoti programinės įrangos kūrime. Idėja, kad reikalavimai visada būtų pilnai įvykdyti, o po to pilnai

įvykdytas kūrimas, po kurio turi būtų pilnai įvykdyta realizacija, praktiškai yra nereali. Šis metodas nepripažino, kad mokymasis gali vykti vystymosi procese, kai reikia pakeisti ankstesnius rezultatus. Iš tikrųjų, kai kuriuos projektus galima priartinti prie vandens krioklio metodo, atlikus kai kuriuos pataisymus.

Iteracijų programinės įrangos kūrimo gyvavimo ciklas numato nuoseklų dėmesį kiekvienai problemos sprendimo fazei, tačiau leidžia programuotojams grįžti atgal ir pakartoti seką, kad pasiekti kiekvienos veiklos rezultatus (Wautelet, Kolp, Poelmans, 2013). Iteracinis programinės įrangos kūrimo gyvavimo ciklas yra labiau artimas realiam pasauliui tuo, kad jis pripažįsta būtinumą grįžti atgal ir padaryti pakeitimus ankstesnėse fazėse, jis pripažįsta, kad programuotojai gali niekada nebaigti kokios nors veiklos. Šiame gryname pavidale iteracinis programinės įrangos kūrimo gyvavimo ciklas mato programuotojus bandančius patikslinti galutinį problemos sprendimą kiekviename cikle. Kadangi dėmesio centre yra galutinis sprendimas, kaip taisyklė, bus reikalinga daug iteracijų, kol jis bus pasiektas, o laikas, kada reikia sustabdyti iteracijas yra labai neapibrėžtas. Augantis iteracinis programinės įrangos gyvavimo ciklas yra dalinis iteracinio atvejis, tikslas pasiekiamas ne pilnai išsprendus problemą, o suskaidomas į mažesnės apimties tikslus. Iteracijos paprastai tęsiasi iki tol, kol ši mažesnė problema nėra pilnai išsprendžiama, o ją išsprendus, iteracijos tęsiamos kitos problemos dalies sprendimui. Kaip ir iteraciniame programinės įrangos kūrimo gyvavimo cikle, augančiame iteraciniame negalima konkrečiai pasakyti apie iteracijų trukmę atliekant konkrečią problemos sprendimo veiklą – tiesiog tam tikra analizė, tam tikri reikalavimai, tam tikras kūrimas, tam tikras realizavimas kartojasi bendroje sekoje. P. Ragunath (2010) teigimu, augantis iteracinis programinės įrangos gyvavimo ciklas kol kas yra geriausia praktika programinės įrangos kūrimui.

Evoliucinių prototipų programinės įrangos gyvavimo ciklas apima augantį iteracinį, tačiau naudojamas tais atvejais, kai galutinis kūrimo tikslas nėra aiškiai apibrėžtas. Tai tokie atvejai, kai kūrėjai ir suinteresuotos šalys nagrinėja galimą sprendimą. Kūrėjai naudoja augančių iteracijų būdą prototipo sukūrimui, o vėliau ieško grįžtamojo ryšio siekdami imtis tolimesnio vystymo galimai kita kryptimi. Akcentuojamas greitas idėjos realizavimas, galimai su klaidomis, jis testuojamas ir nustatoma ar jį verta toliau vystyti. Prototipo sąvoka reiškia, kad kai galutinis tikslas randamas, prototipas atmetamas ir kuriama panaši sistema, didelį dėmesį kreipiant į kokybę. Tačiau dažnai tai pakankamai sunku padaryti kai klientas matė (ir pasiruošęs mokėti bei naudoti) veikiančią programinę įrangą.

Sparčiai besivystančios informacinės technologijos, intensyviai plėtojama IRT infrastruktūra sąlygoja tai, kad kuriamos programinės įrangos sudėtingumas pasaulyje sparčiai auga. Yra sukurta daug PĮ kūrimo modelių, kurie aprašo PĮ kūrimo principus ir tokiu būdu formalizuoja PĮ kūrimo procesus. Vienas geriausiai žinomų yra klasikinis „krioklio“ arba nuoseklusis PĮ kūrimo metodas. Tačiau seni PĮ kūrimo modeliai sunkiai susidoroja su šiomis dienomis PĮ keliamais reikalavimais (France, Rumpe, 2007), todėl kuriami nauji modeliai, labiau atitinkantys šių laikų reikalavimus. Programinės įrangos kūrėjai nuolat ieško būdų kaip patobulinti PĮ kūrimo procesą, kad ją būtų galima kurti greičiau, lengviau, pigiau bei geriau. Vienas tokių yra paskutiniu metu išpopuliarėjęs Agile PĮ kūrimo metodas (Popli, Chauhan, 2013).

Programinės įrangos kūrimo lanksčių metodologijų esmė yra operatyvaus grįžtamo ryšio gavimas ir, kaip pasekmė – neskausminga reakcija į projekto pokyčius (prioritetuose, darbų sąraše, naujose užsakovo idėjose). Tai realizuojama kūrimo proceso transformavimas į trumpas iteracijas, kurios paprastai trunka kelias savaites. Komandos darbo rezultatas kiekvienoje iteracijoje atrodo kaip programinis produktas miniatiūroje ir apima planavimą, reikalavimų analizę, projektavimą, kodavimą ir dokumentavimą. Kiekvienos iteracijos pabaigoje komanda atlieka savo rezultatyvumo įvertinimą ir planuoja darbą sekančiam etapui. Tokios metodologijos turi eilę privalumų, kurių pagrindiniai (McHugh, Conboy, Lang, 2012):

- gera reakcija į pokyčius;
- reguliarus grįžtamasis ryšys iš užsakovo (apie tai, kad daroma tai, ko būtent jis ir nori);
- stabilus ir prognozuojamas rezultatas;
- didelė komandos motyvacija;
- saviorganizavimas;
- rizikų pasidalinimas tarp užsakovo ir komandos.

Gera reakcija į pokyčius. Skirtingai nuo alternatyvios kūrimo metodologijos – waterfall (krioklio), kai užsakovas gauna per iš anksto nustatytą laiką, pavyzdžiui, po metų, griežtai tai, ką jis iš anksto suplanavo projekte, Agile pakeitimus galima įtraukti tada, kai jų reikia, projekto kūrimo procese.

Agile daug dėmesio skiria kokybiškam grįžtamam ryšiui. Dažnai pats užsakovas tiksliai nežino visų savo reikalavimų programiniam produktui ir tuo labiau negali jų suformuluoti. Be to, net labai tiksliai dokumentuotas projektas po pusės metų darbo gali prarasti savo aktualumą ir dažnai susidaro situacijos, kai noras perdaryti nedidelę sudedamąją dalį, kurios pagrindu kuriamas visas programinis produktas, sukelia tai, kad labai ilgai perdaromas galutinis produktas.

Be to, verslo dinamika pati savaime tiesiog nesuteikia laiko detaliam projekto dokumentavimui, nes konkurencinėje aplinkoje yra labai svarbu, kas pirmas paskelbs ir pateiks padarytą projektą. Dokumentavimas reikalauja didelių laiko sąnaudų ir darosi dar sudėtingesnis dėl besikeičiančių užsakovų (vartotojų) pokyčių. Agile numato, kad jau pirmos iteracijos produktas bus baigto funkcionalumo demonstracinė versija, kuriai užsakovas gali pateikti komentarus ir pataisymus praktiškai projekto pradžioje, o po kelių iteracijų bus galima pateikti produkto beta-versiją, kad iš vartotojų gauti grįžtamą ryšį. Agile metodologija leidžia lengvai reaguoti į funkcinių reikalavimų ir prioritetų pokyčius.

Agile užtikrina nuspėjamumą. Agile siūlo komandą suvokti kaip tam tikrą „juodą dėžę“, kuri turi tam tikrą įėjimo duomenų kiekį ir laiką rezultatui pateikti. Iteracijos pradžioje komanda vertina užduotį ir patvirtina tai, kad gali pateikti rezultatą, tačiau nepateikia konkrečių terminų ir įvykdymo kainos – jie kinta kūrimo procese. Ilgalaikio planavimo atsisakymas ir informacijos apie galutinio produkto pateikimo terminus ir kainą nebuvimas gali būti pagrįstas nežymiu kiekiu laiku ir pagal pradžioje numatytą kainą projektų su iš anksto fiksuota kaina.

Pavyzdžiui, jei yra sekantys pradiniai duomenys:

Darbų apimtis = 1000 vnt.

Komandos našumas = 50 vnt. per iteraciją.

Iteracijos kaina (kiekvieno iš jos dalyvių norma) = 100 EUR

Tada rezultate gaunama (1000/50)*100=2000 EUR.

Tai reiškia, kad darbų apimties pasikeitimas 50 vnt. sukelia galutinio produkto vertės pokytį 100 EUR.

Agile leidžia naudoti saviorganizavimą komandos motyvavimui. Saviorganizavimas leidžia nutolti nuo perteklinės vadybos struktūros. Agile numato tik užsakovo atstovą, kuris išreiškia verslo interesus. Be to nėra būtina tikrinti kiekvieną dalyvį: komanda pati pasiskirstys užduotis ir atsakomybę grupės viduje ir būtinai bus garantuota kokybė ir našumas. Tai yra gera našaus komandinio darbo motyvacija.

Agile numato rizikų pasidalinimą tarp komandos ir užsakovo. Šiuo metu labiausiai paplitusios sąveikos su užsakovų rūšys yra kontraktų su fiksuota kaina (fixed-price) arba „laikas ir medžiagos“ (time & material) kontraktų sudarymas. Fiksuoto kainos kontrakto sudarymo atveju visos rizikos, susijusios su projektu, atsakomybė atitenka vykdytojui. Paprastai tai rizikos, susijusios su neteisingais našumo ir sudėtingumo įvertinimais, dėl to, kad šių skaičiavimų pagrindas yra labai silpna dokumentacija. Ypač dideli sunkumai atsiranda tuo atveju, jei projektas

didelis ir apskaičiuotas ne vienam mėnesiui. Po keleto tokių nesėkmių kompanija-vykdytojas, jau patyręs, kad nepilnai paruošė dokumentaciją, ir įvertino neteisingai ir užsakovas ne kartą persigalvojo – pradeda įtraukti didelius rezervus nenumatytoms išlaidoms. Toks kainos padidinimas apsaugo vykdytoją nuo pastovių nukrypimų nuo pradinių reikalavimų, tačiau užsakovas nesuinteresuotas, kad mokėti didesnius pinigus.

Antra kontrakto schema - time & material, kurio esmė yra ta, kad užsakovas apmoka faktinį darbą. Tai sukelia, kad vykdytojas nesiekia prisiimti atsakomybės už rezultatą, o tiesiog neskubėdamas atlieka jam pavestas užduotis. Kaip pasekmė, vykdytojui iš esmės tas pats, ar teisinga kryptimi vykdomas projektas. Užsakovas užmokės ir už laiką, sunaudotą kūrimui neteisinga kryptimi, ir už netinkamų rezultatų perdarymą.

Agile metodologija numato rizikų pasidalinimą, ko rezultate laimi ir užsakovas ir vykdytojas.

Kaip jau buvo pažymėta, Agile turi visą eilę pranašumų prieš kitas programinių priemonių kūrimo metodologijas, tačiau ne visos kompanijos naudoja šias metodologijas, todėl galima išskirti dvi pagrindines priežastis:

- konservatyvumas. Kompanijos vadovybė dažnai klaidingai mano, kad jei yra veikianti schema, kuri neša pelną, tai nėra tikslinga jos keisti. Ir jei tokios organizacijos savo veikloje ir diegia kokias nors naujoves, tai diegia jas minimaliai, siekdamos nepakenkti tam, kas veikia. Tokia pozicija gera trumpalaikiu laikotarpiu, tačiau ilgalaikėje perspektyvoje ji gali sukelti technologinį ir metodologinį vykdytojų atsilikimą ir, kaip pasekmę – kuriamos programinės produkcijos konkurencingumo praradimą;
- nesėkminga diegimo patirtis. Kai kurioms komandoms tiesiog nesigauna naudotis nauja metodologija. O dažnai atsiranda situacija, kai buvo bandyta diegti metodologiją ar net dalinai ji buvo įdiegta, tačiau laukiamo rezultato nebuvo.

Visų pirma, prieš pereinant prie Agile metodologijos, reikia kruopščiai paruošti dirvą diegimui tiek projekto lygyje, tiek ir vadovybės lygyje. Būtent tam reikalingi profesionalūs mokymai, kurie gali ne tik paruošti komandą naujovėms, bet ir kontroliuoti situaciją. Mokymai gali padėti efektyviai:

- paruošti darbuotojus naujovėms;
- paruošti ir apmokyti kvalifikuotą personalą;
- apmokyti metodologijos;
- optimizuoti procesus kompanijoje;

- gauti ekspertinį palaikymą ir konsultavimą sudėtinguose momentuose.

Mokymai gali padėti konkrečiuose metodologijos diegimo klausimuose

1.2. IT projektų valdymas

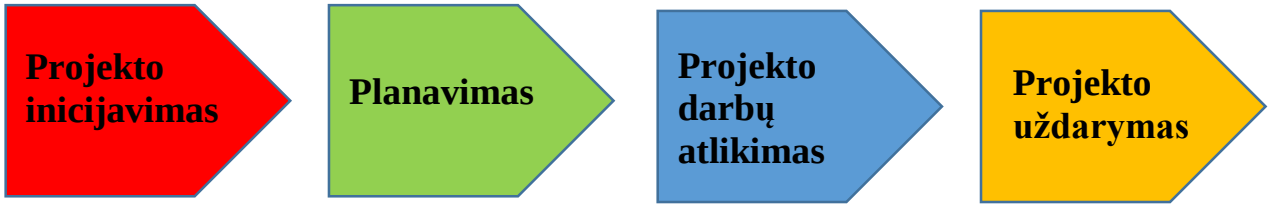
Vystantis informacinėms technologijoms viena organizacijų dalis pradėjo automatizuoti savo veiklą, o kita – pradėjo kurti programines priemones, kuri tiekiama pirmojo tipo organizacijoms. Ir pirmu ir antru atveju organizacijos dirba su IT projektais.

Sąvoką „IT projektas“ galima aprašyti bendru „projekto“ sąvokos pavidalu, tačiau su tam tikrais papildymais, visų pirma, susijusiais su tuo, kad IT projektas numato veiklą, nukreiptą į informacinių technologijų kūrimą ir panaudojimą (Dalcher, 2013). Reikia pažymėti, kad IT projektą nuo paprasto projekto, visų pirma, skiria tai, kad IT projekto atveju yra disonansas tarp užsakovo ir vykdytojo. Dažniausiai IT projektus užsako organizacijos verslo skyrius, o projektą diegia savo IT skyrius arba savarankiška IT kompanija. Esant tokiai situacijai akivaizdu, kad galimi sunkumai nustatant kuriamos ir diegiamos programinės priemonės verslo reikalavimus ir lūkesčius, nes IT sritis yra gana specifinė. Tai reiškia, kad pirmuose etapuose tarp užsakovo ir vykdytojo vykstančiuose aptarimuose aiški idėja gali būti ir nesuformuluota. Šiuo atveju techniniame projekte fiksuojamos pakankamai neaiškiai išreikštos idėjos, kurios vėliau keičiamos. Užsakovui pirmą kartą skaitant techninę užduotį, jam gali kilti eilė klausimų, ir tai natūralu, nes techninės užduoties kūrėjas kai kuriuos IT projekto aspektus galėjo suprasti gana subjektyviai. Kitas IT projekto ypatumas yra tolygus kaltės paskirstymas projekto dalyviams, jei jis nesėkmingai realizuojamas. Tokiu būdu kaltė už rezultatą tenka ne tik vykdytojui (už galimą neteisingą realizaciją) arba ne tik užsakovui (už galimą klaidą formuluojant reikalavimus), o visiems dalyviams kartu todėl, kad vienaip ar kitaip jie turi sukurti efektyvią tarpusavio komunikaciją, minimizuojant subjektyvumą ateityje. Trečias IT projekto ypatumas yra jo didelė kaina. Jei statant pastatą galimi tik minimalūs nukrypimai nuo galutinių reikalavimų ir lūkesčių, tai IT projektuose yra labai tikėtina. Yra esminių pakeitimų įtraukimo rizika, o kadangi labai dažnai santykiai tarp užsakovo ir vykdytojo reguliuojami anksčiau pasirašytu kontraktu, tai bet koks pakeitimas sukelia taip vadinamą „užklausą pakeitimui“, kuri kainuoja tam tikrą pinigų sumą. Didelis pakeitimų skaičius numato ir didelę pinigų sumą, todėl IT projektai laikomi vienais iš brangiausių projektų.

Kiekvienas IT projektas nuo jo sukūrimo momento iki pabaigos, kai informacinė technologija pilnai sukurta ir įdiegta, pereina keletą tarpinių etapų. Kitaip sakant, projektas turi fazes, kurios bendrai vadinamos projekto gyvavimo ciklu. Projekto gyvavimo ciklas - tai, kaip

taisyklė, nuoseklių ir kartais persidengiančių, projekto fazių, kurių pavadinimai ir skaičius apibrėžiami organizacijos ar organizacijų, įtrauktų į projektą, valdymo ir kontrolės poreikiais, paties projekto pobūdžiu ir jo taikymo sritimi (PMBOK, 2008).

Gyvavimo ciklas padeda koordinuoti projekto dalyvių pastangas, nes jis įvertina tam tikrą specifiką. Pavyzdžiui, projekto vadovas turi suprasti, kokią įtaką daro vieno etapo įėjimai ir išėjimai kitiems gyvavimo ciklo etapams. Projekto gyvavimo ciklas tiesiogiai susijęs su jo finansavimu, nes tik suplanavus jo realizacijos fazes, galima prognozuoti kainą ir sąnaudas.



1 pav. IT projekto gyvavimo ciklo schema (PMBOK, 2008)

Bendrais bruožais IT projekto gyvavimo ciklą sudaro keturios fazės: inicijavimas, planavimas, vykdymas ir uždarymas (1 pav.):

- inicijavimas: sukuriama projekto idėja, apibrėžiami verslo reikalavimai ir tikslai;
- planavimas: sudaromi projekto planas, finansinis planas, kokybės planas ir išteklių planas, apibrėžiamas darbas, kuris turi būti atliktas projekto rėmuose;
- vykdymas: užduočių atlikimas; gautus rezultatus vertina projekto vadovai ir pasekmės lyginamos su projekto planu;
- uždarymas: galutinis rezultatas ir suformuota dokumentacija pateikiami užsakovui; formuojamas projekto sėkmingumo lygis.

Apibendrinant galima pasakyti, kad projekto gyvavimo ciklas turi pradinį ir pabaigos taškus, susietus su laiko skale. IT projekto gyvavimo ciklas apima visas fazes, pradedant inicijavimu ir baigiant projekto užbaigimu. Perėjimai nuo vienos fazės prie kitos paprastai nėra aiškiai apibrėžti, išskyrus tuos atvejus, kad jie formaliai atskiriami pasiūlymo priėmimu ir leidimo darbui tęsti gavimu. Tačiau inicijavimo fazės pradžioje dažnai atsiranda sunkumai tiksliai nustatant momentą, kada darbą jau galima identifikuoti kaip projektą, ypač jei kalbama apie IT projektus.

2. PROGRAMINĖS ĮRANGOS KŪRIMO NAUDOJANT AGILE PROJEKTŲ VALDYMO METODUS TEORINIAI ASPEKTAI

2.1. Programinės įrangos kūrimas naudojant Agile metodus teoriniu aspektu

Šiuo metu programinės įrangos kūrimas yra techniškai sudėtingesnis, labiau strateginis ir į pirmą vietą iškelia skirtingus platesnio suinteresuotų šalių rato požiūrius (Nerur ir kt., 2010). Programinės įrangos kūrimas yra kompleksinė veikla, kuri charakterizuojama uždaviniais ir reikalavimais, pasižyminčiai aukštu kintamumo laipsniu (Nerur, Mahapatra, Mangalaraj, 2005). Tai neišvengiamai leidžia padaryti išvadą, kad programinės įrangos kūrimas dabar ir ateityje bus vykdomas vis labiau neapibrėžtame kontekste (Nerur ir kt., 2010).

Agile metodai atsirado tam, kad išspręsti daugelį šių problemų. Šie metodai buvo priimti ir prijungti prie pagrindinių programinės įrangos kūrimo metodų (Kettunen, Laanti, 2008). H. Erdogmus (2009) teigimu, Agile programinės įrangos kūrimo metodas yra svarbiausia paradigma, kuri per paskutinį dešimtmetį apėmė visą programinės įrangos kūrimo sritį. Net jei šis metodas ir netapo populiariausias programinės įrangos kūrimo metodu tarp visų naudojamų, jis tapo vienu iš labiausiai aptariamų programinės įrangos kūrimo metodų. Šio metodo žodynas ir pagrindinės idėjos jau persikėlė į kitas sritis, o ypatingai į projektų valdymo sritį. Taip pat minėtas autorius pripažįsta, kad Agile metodas yra labai įvairiapusis ir jo traktavimas priklauso nuo asmeninių ir kontekstinių interpretavimų. Tai Agile metodą daro sunkiai tiriamu, nes mokslininkai stengiasi savo tyrimuose išvengti klaidingų interpretavimų.

Agile metodų panaudojimas žada aukšto našumo pasiekimą ir aukštos programinės įrangos kokybės užtikrinimą, kas pritraukia įmonių, kurios reikalauja vis didesnio vystymosi greičio ir aukštesnės jų produkcijos kokybės. Kaip taisyklė, išskiriami sekantys Agile metodų privalumai (Leffingwell, 2011):

- padidėja vartotojų pasitenkinimas;
- sutrumpėja produkto pateikimo rinkai laikas;
- padidėja kokybė;
- pagerėja projektų portfelis ir produkto valdymas (projektų tipai, savybės);
- pagerėja produktų kūrimo investicijų valdymas (kontrolė ir lankstumas)
- sumažėja nuostoliai (padidėja efektyvumas, našumas, sumažėja kūrimo kaštai);
- lengvesnis ir tikslesnis būsimos situacijos prognozavimas;
- pagerėja rizikų valdymas (rizikų mažinimas);

- geresnis produkto kūrėjų pasitenkinimas.

Tačiau reikia pažymėti, kad nežiūrint į tai, jog empirinių duomenų apie šiuos pranašumus kiekis didėja, jų vis dar nepakanka, kad šie pranašumai būtų pilnai patvirtinti (Dybå, Dingsøyr, 2008). Nežiūrint į šiuo privalumus, Agile metodai susiduria ir su tipinėmis programinės įrangos kūrimo problemomis, tarp kurių galima paminėti pateikimo rinkai terminų spaudimas, našumo reikalavimai, nevienareikšmiški ir neaiškiai apibrėžti reikalavimai produktui, greitai besikeičianti aplinka (Baskerville ir kt., 2006).

Agile metodas atsirado apie 2000 metus, tačiau turi gilias šaknis programinės įrangos kūrimo istorijoje. Agile idėjas galima atrasti F. Brooks (1975) ir J. Naumann, A. Jenkins (1982) darbuose. Agile meodas susiformavo kaip eilės veiksnių, idėjų ir geriausių praktikų pasekmė, kurios atsirado objektiškai-orientuoto programavimo vystymosi kontekste. Šios idėjos pakeitė programinės įrangos kūrimo būdą ir pačių kūrėjų mąstymą (Abbas, Gravell, Wills, 2008). Kadangi į šias idėjas tuo metu nebuvo kreipiama daug dėmesio, reikėjo apie 30 metų, kad jos būtų pripažintos kaip efektyvus programinių priemonių kūrimo būdas (Corbucci ir kt., 2011).

Paskutinio praėjusio amžiaus dešimtmečio gamybos veiksnių kombinacija paruošė puikią dirvą Agile idėjų vystymuisi (Abbas, Gravell, Wills, 2008). Visų pirma, tai buvo reakcija į sudėtingus, griežtai reglamentuojamus programinės įrangos kūrimo būdus. Visų antra, didėjantis pokyčių verslo aplinkoje lygis reikalavo, kad programuotojai apdorotų sudėtingus ir nenuspėjamus reikalavimus sistemų vystymuisi. Tokiu būdu programinės įrangos kūrimo praktikai pradėjo analizuoti alternatyvius programinės įrangos kūrimo būdus, tokius kaip Interaktyvus kūrimas, glaudžiai bendradarbiaujant su užsakovu (Royce, 1987), nepertraukiamos integracijos metodas (Larman, Basili, 2003), greitas prototipavimas (Naumann, Jenkins, 1982). Antroje XX amžiaus paskutinio dešimtmečio pusėje moksliniai tyrimai ir praktiniai rezultatai objektiškai-orientuoto programavimo, šablonų kūrimo, automatinio testavimo srityse parengė bendrą mąstymo kryptį, kurios pagalba buvo parengta eilė programinės įrangos kūrimo metodų, grindžiamų Agile principais. Tai tokie metodai, kaip ekstremalus programavimas (XP), dinaminis sistemų kūrimo metodas (DSDM), adaptyvus programinės įrangos kūrimo metodas, Crystal metodas, savybėmis pagrįstas kūrimas, pragmatinis programavimas ir kt.

2001 metais grupė nepriklausomų praktikų, turinčių stiprias sąsajas su programinės įrangos kūrimo industrija ir silpnesnes, tačiau aktualias, sąsajas su akademinių sluoksnių tyrimo grupėmis nusprendė apjungti savo jėgas ir įkūrė tai, kas vėliau buvo pavadinta Agile judėjimu. Tam, kad šios idėjos būtų konkretizuotos, 17 programinės įrangos ekspertų kolektyviškai parengė Agile

manifestą (Manifesto for Agile Software Development, 2001). Manifesto tikslas buvo atkreipti dėmesį į tai, kad norint sukurti aukštos kokybės programinę įrangą, kūrėjų komanda turi koncentruotis į: 1) žmones ir jų bendravimą; 2) veikiančią programinę įrangą; 3) bendradarbiavimą su klientu ir 4) reagavimą į neplanuotus pokyčius. Šios sritys buvo pažymėtos kaip svarbesnės, nei procesai ir įrankiai, išsami dokumentacija, derybos dėl kontraktų ir plano vykdymas. Agile metodai buvo reakcija į labiau tradicinius, žinomus kaip paremtus planavimu. Agile yra taip pat žinoma ir kaip pokyčiais paremta kūrimo paradigma (Moe, 2011).

Remiantis T. Stober ir U. Hansmann (2011), Agile metodas nėra apibrėžiamas nedideliu praktikų ir technikų rinkiniu. Agile metodas apibrėžia strateginį potencialą, galimybę kurti ir reaguoti į pokyčius, galimybę subalansuoti lankstumą ir struktūrą, galimybę pritraukti kūrėjų komandos kūrybiškumą ir inovacijas, o taip pat galimybę veikti kintamumo ir neapibrėžtumo sąlygomis.

S. Nerur ir kt. (2010) pateikė papildymą, liečiantį Agile metodų naujumą. Tokios sąvokos kaip kūrimas iteracijomis, mokymas, saviorganizavimas, refleksyvi praktika, suinteresuotų šalių dalyvavimas, vystėsi atskirai kitose disciplinose, tačiau viena ar kelios iš jų tam tikra forma buvo naudojamos ir programinės įrangos kūrime. Tai taip pat gali būti panaudota tam, kad praktiškai būtų galima vystyti ir valdyti programinės įrangos kūrimo komandas ir projektus.

Agile būdo priėmimas sukelia pokyčius organizacijoje, tame tarpe jos struktūroje, kultūroje ir valdymo praktikoje, darbo procedūrose, įrankiuose ir technikose, komunikacijos kanaluose, problemų sprendimo strategijose ir žmonių vaidmenyse (Moe, 2011).

Racionalus, inžinerinis programinės įrangos kūrimo būdas dominavo praktiškai nuo jo sukūrimo (Nerur, Mahapatra, Mangalaraj, 2005). Platus išankstinis planavimas yra pagrindas problemų ir pokyčių prognozavimui, matavimui ir valdymui viso kūrimo gyvavimo ciklo laikotarpiu. Tradicinis programinės įrangos kūrimo būdas yra orientuotas į procesą, vadovaujasi įsitikinimu, kad pokyčių šaltinis gali būti identifikuotas ir pašalintas naudojant pastovius matavimo ir koregavimo procesus (Popli, Chauhan, 2013). Pagrinde akcentuojamas optimizavimo ir pasikartojančių procesų atlikimas, kuriuose planavimas ir kontrolė vykdomi panaudojant komandinį ir kontroliavimo valdymo stilių.

Remiantis tradiciniu būdu, programinės įrangos kūrimas, kaip taisyklė, vadovaujasi gyvavimo ciklo modeliais, tokiais, kaip krioklio, spiralės ar jų variacijomis (Nerur ir kt., 2010). Šie modeliai apibrėžia uždavinius ir su jais susijusius vaidmenis bei pageidaujamus kiekvienos fazės rezultatus. Kaip priedą prie galutinio produkto kodo, šie metodai taip pat reikalavo suformuoti

didelį kiekį dokumentacijos, kurioje įtvirtinamos žinios apie procesą ir produktą. Ryšiai tarp projekto dalyvių taip pat formalizuoti per šiuos dokumentus (nors gali būti ir kitokių būdų). Vartotojai vaidina svarbų vaidmenį specifikavime ir patikrinime, tačiau jų dalyvavimas yra minimalus kitose veiklose (Nerur ir kt., 2010). Tradiciniai būdai numato vieno ciklo ar adaptyvų mokymą, kurio koncepcijoje problemos sprendimo mechanizmai apibrėžia tikslo paieškos būdą.

Kita vertus, Agile metodai taikomi neapibrėžtumo sąlygomis ir remiasi žmonių kūrybiškumu, o ne procesais (Cockburn, Highsmith, 2010). Jie charakterizuojami trumpais, pasikartojančiais prioritetinių atliekamų funkcijų ciklais, savianalizės ir apmastymo periodais, bendru sprendimų priėmimu, greito grįžtamo ryšio ir pokyčių įtraukimu, o taip pat nepertraukiamu programinės įrangos kodo pakeitimų integracija kūrimo stadijoje. Komanda dirba su nedidelėmis produkto dalimis, kiekviena iš kurių apima planavimą, kūrimą, integravimą, testavimą ir pateikimą. Programinio produkto kūrėjai dirba su klientais, kurie yra aktyvūs kūrimo proceso dalyviai, kartu vykdo prioritetines funkcijas, kas leidžia bendrai priimti sprendimus.

Agile metodai skatina lyderystės ir bendradarbiavimo valdymo stilių, kur projekto vadovo vaidmuo yra tas pats kaip ir tarpininko ar koordinatoriaus vaidmuo (Nerur ir kt., 2010). Kiekviename kūrimo cikle vartotojui stengiamasi pateikti naudingą veikiantį programinį kodą. Agile metodai užkerta kelią didelėms dokumentacijoms, nesusijusioms su kodu, apimtimis. Tokiu būdu projekto žinios tampa suprantamomis be žodžių. Komandos narių rotacija garantuoja, kad šios žinios nepriklausys tik keliems žmonėms. Agile metodai pasižymi didele socialine sanglauda, kurioje platus bendradarbiavimas ir ryšiai užtikrina pagrindą kolektyviniams veiksmams. Įvairios suinteresuotos šalys ir galutiniai vartotojai pereina per pasikartojančius mintis-veiksmas-refleksija ciklus, kurie skatina mokymosi ir adaptacijos aplinką. Toks pastovus prieštaravimų egzistuojančioms normoms ir vertybėms vertinimas skatina taip vadinamą „dvigubą mokymo ciklą“, kuris yra „generuojantis“, didindamas tiek mokymąsi, tiek ir gebėjimą inovacijoms ir panaudoja vieno iš jų pranašumus (Ford, Voyer, Wilkinson, 2010). Komandos nariams suteiktos didesnės sprendimų priėmimo galios ir jie neapsiriboja specializuotu vaidmeniu, kas sudaro sąlygas saviorganizacijai ar atsakui naujose netikėtose situacijose (Nerur ir kt., 2010).

Kaip taisyklė, tradiciniai programinės įrangos kūrimo proceso tobulinimo būdai palaiko universalaus ir pasikartojančio programinės įrangos kūrimo ir tobulinimo ideologiją. Tradicinių tobulinimo būdų iniciatyvos didele dalimi yra kontroliuojamos organizaciniame ir valdymo lygmenyje (Nerur, Mahapatra, Mangalaraj, 2005). Tačiau programinės įrangos Agile kūrimo metodai reikalauja, kad reguliariais intervalais komanda apmastytų kaip tapti labiau efektyviais, ir

atitinkamai reguliuoja savo elgseną (Salo, Abrahamsson, 2007). Pagrindiniai tradiciniai tobulinimo metodai nukreipti palaikyti tobulinimą organizaciniame lygyje, siekiant atlikti geresnius programinius projektus. Kita vertus, Agile metoduose, pagrindinis adaptacijos proceso dėmesys Agile projekto komandose yra skiriamas greitam programinės įrangos kūrėjų patirties panaudojimui, pastoviai gerinant vykdomą procesą (Salo, Abrahamsson, 2007).

1 lentelėje pateikti pagrindiniai teoriniai skirtumai tarp tradicinių ir Agile metodų.

1 lentelė. Pagrindiniai teoriniai skirtumai tarp tradicinių ir Agile metodų (Salo, Abrahamsson, 2007)

	Tradiciniai PĮ kūrimo metodai	**Agile PĮ kūrimo metodai**
Pagrindinės prielaidos	Sistema yra pilnai specifikuojama, nuspėjama ir gali būti sukurta, panaudojant kruopštų ir platų planavimą	Mažos komandos, naudojančios nepertraukiamą` kuriamos PĮ tobulinimą gali sukurti aukštos kokybės, adaptyvią PĮ ir vykdyti testavimą, greito grįžtamo ryšio ir pakeitimų pagrindu
Valdymas	Orientuotas į procesus	Orientuotas į žmones
Žinių valdymas	Aiškus	Neišreikštas
Vaidmenų priskyrimas	Individualus: pirmenybė specializacijai	Saviorganizuojančios komandos: skatina tarpusavio vaidmenų apsikeitimą
Vartotojo vaidmuo	Svarbus	Kritinis
Pageidaujama organizacinė forma/struktūra	Mechanistinė (biurokratinė su aukštu formalizavimu)	Organinė (lanksti ir dalyvavimu skatinanti bendrus socialinius veiksmus)
Kūrimo procesas	Apgalvotas ir formalus, tiesinė žingsnių seka, atskiras formulavimas ir įgyvendinimas, pagrįstas taisyklėmis	Besivystantis, pasikartojantis ir tiriamasis, žinių ir veiksmų neatskiriamumas, nėra formalių taisyklių
Tikslas	Optimizavimas	Adaptavimas, lankstumas, atsakomoji reakcija
Problemų sprendimo būdai	Geriausių priemonių parinkimas, kad pasiekti galutinį tikslą gerai suplanuotomis ir formalizuotomis priemonėmis	Mokymasis eksperimentuojant ir analizuojant save, pastovus problemos performavimas ir jos sprendimas
Požiūris į aplinką	Stabili, prognozuojama	Kintanti, sunkiai nuspėjama
Mokymosi tipas	Vieno ciklo, adaptyvus	Dvigubo ciklo, generuojantis

Pagrindinės charakteristikos	Kontrolė ir nukreipimas Konfliktų vengimas Inovacijų formalizavimas Vadovas yra kontroliuojantis asmuo Dizainas numatomas prieš realizavimą	Bendradarbiavimas ir komunikavimas Apima konfliktus ir dialektiką Siekia tyrimo ir kūrybiškumo Vadovas yra koordinatorius Kūrimas ir diegimas neatskiriami
Dėmesys į procesų tobulinimą	Organizacinių programinės įrangos kūrimo procesų tobulinimas (ateities projektams)	Nuoseklus kasdieninės darbo praktikos tobulinimas viso projekto metu
Teorinis pagrindas	Loginis pozityvizmas, mokslinis metodas	Mokymosi teorija, pragmatizmas, fenomenologija

Agile konceptas gamybos srityje apima galimybę pateikti naujus produktus rinkai ir reaguoti į klientų poreikius greitai, įtraukiant inovacines valdymo struktūras, lanksčias technologijas ir gabių darbuotojų žinių bazę (Kettunen, 2009). Mokslinėje literatūroje galima rasti visą eile Agile programinės įrangos kūrimo apibrėžimų. Literatūra pateikia neryškius Agile programinės įrangos kūrimo pagrindus ir charakteristikas, tačiau bendra visiems apibrėžimams yra tai, kad Agile apibrėžiamas reagavimo į pokyčius sąvokomis (Lee, Xia, 2010). Agile programinės įrangos kūrimo apibrėžimai pateikiami 2 lentelėje.

2 lentelė. Agile programinės įrangos kūrimo apibrėžimai

Autorius	Apibrėžimas/konceptas/idėja
K. Conboy, B. Fitzgerald (2004)	Agile programinės įrangos kūrimas yra pastovus objekto pasiruošimas greitam, aktyviam ar reaktyviam pokyčių priėmimui dėka aukštos kokybės, paprastų, ekonominių komponentų ir ryšių su aplinka
J. Highsmith (2004)	Agile programinės įrangos kūrimas – tai gebėjimas ir kurti, ir reaguoti į pokyčius, siekiant gauti pelną besikeičiančioje verslo aplinkoje; tai gebėjimas subalansuoti lankstumą` ir stabilumą
C. Larman, V. Basili, V. (2003)	Agile programinės įrangos kūrimas yra greitas ir lankstus atsakas į pokyčius.
J. Erickson, K. Lyytinen, K. Siau (2005)	Agile programinės įrangos kūrimas yra susietas su tokiais konceptais, kaip vikrumas, lankstumas, greitis, budrumas; tai reiškia atmetimą sunkių tradicinių programinės įrangos kūrimo metodų, kad būtų galima greitai reaguoti į besikeičiančios aplinkos pokyčius ir besikeičiančius vartotojų reikalavimus.
B. Henderson-Sellers, M. Serour (2005)	Agile programinės įrangos kūrimas yra susijęs su pasiruošimu veiksmui ar pokyčiui; jis turi dvi dimensijas: (1) galimybę priimti įvairius pokyčius ir (2) galimybę greitai prisiderinti ar pertvarkyti PĮ kūrimo procesus, kai tai būtina

K. Lyytinen, G. Rose (2006)	Agile PĮ kūrimas apibrėžiamas kaip gebėjimas greitai suvokti ir reaguoti į techninius pokyčius ir naujas verslo galimybes; jis numato tyrimais paremtą mokymąsi ir bandymais paremtą mokymąsi
A. Cockburn (2006)	Agile PĮ kūrimas yra lengvas, beveik pakankamas ir manevringas
A. Qumer, B. Henderson-Sellers (2008)	Agile PĮ kūrimas – tai atkaklus subjekto elgesys ar gebėjimas parodyti lankstumą priimant tikėtinus ar netikėtus pokyčius greitai, su mažiausiais laiko nuostoliais, paprastais ir kokybiškais instrumentais dinamiškoje aplinkoje. Agile PĮ kūrimas gali būti įvertintas lankstumu, greičiu, mokymusi ir reagavimu į pokyčius.
D. Greer, Y. Hamon (2011)	Agile PĮ kūrimas siekia padidinti kokybės ir aptarnavimo veiksnius pagal einamus vartotojo reikalavimus, lanksčiai reaguojant į vartotojo poreikius ir rinkos pokyčius.
S. Ahalt ir kt. (2014)	Agile PĮ kūrimas yra pastovus pasiruošimas greitai, reaktyviai ar proaktyviai priimti pokyčius ir mokytis iš pokyčių, tuo pačiu didinant vartotojui kuriamą vertę (ekonomija, kokybė ir paprastumas).

Apibendrinant 2 lentelėje pateiktus apibrėžimus galima pasakyti, kad lankstumas yra būtina Agile programinės įrangos kūrimo sąlyga. Tačiau vien tik lankstumo Agile programinės įrangos kūrimui nepakanka. Agile programinės įrangos kūrimas apima skirtingų pokyčių tipus į kuriuos reaguojama efektyviai, didelio greičio ir mažų kaštų požiūriu (Qumer, Henderson-Sellers, 2008). Taip pat apibrėžimuose galima rasti ir tokias dimensijas, kaip tyrimai ir eksploatacija pagrįsti mokymai, o taip pat mokymasis iš pokyčių.

Remiantis N. Kurapati, V. Manyam ir K. Petersen (2012) Agile programinės įrangos kūrimas gali būti apibrėžtas tokio kūrimo būdo atributais: lankstumas, greitis, taupumas, reagavimas ir mokymas. Remiantis minėtais autoriais:

Lankstumas – tai ūkio subjekto gebėjimas ar elgesys, leidžiantis adaptuotis prie besikeičiančios aplinkos sąlygų tada, kai to reikia. Metodas ar metodo fazė gali demonstruoti lankstumą prisiderindama prie tikėtinų ar netikėtų pokyčių. Tai programinės įrangos kūrimo metodo galimybė sukurti pokyčius, priimant išorinės aplinkos pokyčius per vidinius komponentus.

Objekto **greitis** charakterizuoja greitą elgseną, kad pasiekti reikiamą tikslą. Greitas metodas gali padėti pateikti rezultatą, panaudojant ypatingus veiklos būdus.

Taupumas remiasi kompaktiškumu ir tvarkingumu. Taupus metodas leidžia pasiekti kokybišką rezultatą ekonomiškai, per trumpiausią galimą laiką, taikant paprastas ir kokybiškas

programinės įrangos kūrimo priemones. Tai vertės maksimizavimas, taikant ekonomiją, kokybę ir paprastumą.

Mokymąsi sudaro žinios bei tobulėjimas ir yra neprasiejamas subjekto gebėjimas, kuris, visų pirma, pasiekiamas savalaikių žinių ir patirties, sukauptų ankstesnėse praktikose, pagalba. Mokymosi metodas yra pastovus tobulėjimas.

Reagavimas reiškia būtinumą atsakyti, kai to reikalauja situacija. Tai reiškia, kad neužtenka tik priimti pokyčius. Pokyčiai turi būti realizuoti ir aiškūs.

Remiantis šiomis Agile dimensijomis reikalinga įvertinti Agile programinės įrangos kūrimo metodo įtaką komandos darbo efektyvumui.

2.2. Efektyvumo samprata

Remiantis įvairiais literatūros šaltiniais, galima pasakyti, kad veiklos efektyvumo apibrėžimų yra išties nemažai, kurie daugiau ar mažiau atskleidžia jo esmę, tačiau jo reikšmė Lietuvoje ne visada tinkamai suprantama.

Efektyvumo terminas yra kilęs iš lotynų kalbos žodžio „effectus", reiškiančio vykdymą arba veiksmą. Ankščiau, efektyvumo sąvoka buvo siejama tik su technika it technologijomis ir buvo suprantama kaip darbo mato santykis panaudotos energijos atžvilgiu arba darbo santykis su faktiniu ir potencialiu bet kokios veiklos rezultatu. Rinkos ekonomikos terminų žodynėlyje efektyvumas apibrėžiamas kaip gamybos išteklių panaudojimo būdas, garantuojantis maksimalų produktą su minimaliomis sąnaudomis. Tačiau efektyvumo sąvoka nėra gryna objektyvumo savybė, o neišvengiamai priklauso nuo įvertinimo ir yra vertinimo kategorija. (Richard, ir kt., 2009).

Puškorius (2002) teigia, kad efektyvumas – tai santykis tarp pageidautinų veiklos rezultatų ir panaudotų tiems rezultatams pasiekti kompleksinių išteklių, indėlių, išlaidų bei kitų resursų. Vainienė (2008) efektyvumą apibrėžia panašiai, kita vertus jos apibrėžime akcentuojamas sąnaudų minimizavimas, siekiant rezultato maksimizavimo: efektyvumas - tai išteklių panaudojimo veiksmingumas, kai norimas rezultatas pasiekiamas mažiausiomis įmanomomis sąnaudomis arba naudojant turimus išteklius pasiekiamas maksimalus įmanomas rezultatas. Jeigu ekonomiškume yra vertinami tik kaštai, tai efektyvumo rodiklis yra daug universalesnis, čia yra vertinami ir ištekliai, ir kaštai, ir indėliai ir visi kiti resursai. Mackevičiaus (2007) nuomone efektyvumas suprantamas kaip racionalus lėšų gamybos procese cirkuliavimas, duodantis teigiamą gamybos rezultatą, greitą gamybos proceso ciklą, kurio metu ne tik sukuriamas pelnas, bet ir pinigų srautas, reikalingas gamybos proceso tęstinumui palaikyti. Galima šiame apibrėžime pastebėti tai, kad

įmonė, siekdama pagerinti ekonominį efektyvumą, turi siekti gauti kuo didesnį pelną, nes jo pakankamumas leis daryti investicija, diegti inovacijas, plėsti savo veiklą, kurti naujus produktus ir tobulinti esamus, gerinti jų kokybę. Tačiau gauti didelį pelną įmonėms dažnai sutrukdo per didelės išlaidos, o jų optimizavimas susijęs su veiksmingumu, ekonomiškumu ir efektyvumu. Taip pat galima pastebėti, kad įmonė gali veikti efektyviai, jei taupomi ištekliai ir vis dėlto nebūti produktyvia, nes šis rodiklis dėmesį sutelkia į produkciją, o efektyvumas – į sąnaudas, apibrėžiantis šių rodiklių ryšį. Iš kitos pusės produktyvumas/našumas – tai efektyvumo matas, kuris sąnaudas (darbą, kapitalą, medžiagas ir kt.) paverčia produkcija. Iš esmės našumas – tai produkcijos kiekis sąnaudų vienetui.

3 lentelė. Skirtingų autorių efektyvumo apibrėžimai

Autorius	Apibrėžimas
S. Dolan (2006)	Efektyvumas – tai teisingų tikslų pasirinkimas į kuriuos fokusuojama visa energija
P. Drucker (2008)	Efektyvumas – tai ne tik rezultato ryšys su užsibrėžtais tikslais, bet ir rezultatas (efektas) optimalaus išteklių (materialių, finansinių, darbo) panaudojimo požiūriu
M. Mescon (2007)	Efektyvumas – vidinis ekonomiškumas, kuris matuoja geriausią išteklių panaudojimą
T. Pyzdek, P. Keller (2009)	Efektyvumas – tai visada tam tikras santykis (rezultato su tikslais arba rezultato su išlaidomis jam pasiekti), t.y. santykinis dydis, ši kategorija yra vadybinio pobūdžio ir atspindi iškeltų tikslų pasiekimo laipsnį. Efektyvumas – tai gebėjimas atnešti efektą, proceso, projekto ir t.t. rezultatyvumas, kurie apibrėžiami kaip rezultato ir išlaidų, kurios užtikrino šį rezultatą, santykis

Šie apibrėžimai atspindi keturias autorių nuomones. Remiantis Dolan (2006) efektyvumą supranta kaip įmonės veiklos teisingų tikslų pasirinkimą. Drucker (2008) – optimalaus išteklių panaudojimo rezultatas, Mescon (2007) – įmonės išteklių panaudojimo įvertinimą. Populiariausiais yra efektyvumo, kaip rezultato ir išlaidų santykio arba rezultato ir numatytų tikslų santykio apibrėžimai.

Tradiciškai efektyvumas buvo apibrėžiamas kaip pagamintos produkcijos apimčių ir panaudotų išteklių santykis. Remiantis tokiu efektyvumo supratimu, įmonės tikslas – operacinis tobulumas, kai procesai (operacijos) gerinamos išlaidų, išteklių mažinimo sąskaita, kad gauti

standartinį rezultatą. Kad pasiekti tokį tobulumą bet kokie organizacijos pozicijos pokyčiai yra nepageidautini. Remiantis naujų būdu, efektyvumas apibrėžiamas kaip sukurtos vertės ir jai sukurti panaudotų išteklių apimčių santykis (Richard, et al., 2009).

Apibendrinus daugelių autorių požiūrius ir sąvokų interpretacijas galima teigti, kad įmonių veiklos efektyvumo sąvoka apima visas, prieš tai pateiktus pagrindinius įmonės ūkinės veiklos rodiklius, kurie parodo jos rezultatus: veiksmingumą, ekonomiškumą, produktyvumą/našumą, pelningumą, produkcijos, paslaugų ar darbo kokybę, naujovių diegimą. Ekonomiškumas, ne efektyvumas ir ne veiksmingumas, priklauso nuo santykinai mažesnio skaičiaus veiksnių, kurie susiję su lėšomis ir gali būti įvertinti kiekybiškai. Efektyvumo kriterijus nagrinėja daug daugiau veiksnių, išskirtų iš ekonomiškumo, produktyvumo, pelningumo, o veiksmingumas yra aukščiausio lygmens kriterijus, apimantis visų efektyvumo kriterijų visumą ir nustato jų įtaką įmonės veiklos rezultatams bei parodo kokiu efektyvumo laipsniu jie buvo pasiekti.

2.3. Agile metodai programinės įrangos kūrimui

2.3.1. Agile Lean

M. Emiliani (2006) nuomone, Lean programinės įrangos kūrimo metodas pateikia principų, skirtų programinės įrangos kūrimo proceso analizei ir pagerinimui, šlamšto minimizavimo ir vertės vartotojui maksimizavimo požiūriu, rinkinį. Lean programinės įrangos kūrimo metodo taikymas privertė daugelį įmonių peržiūrėti savo programinės įrangos kūrimo praktikas, gerokai pagerinant produktų kokybę, sumažinant jų kūrimo sąnaudas ir pagreitinant jų kūrimo procesą. Todėl programinę įrangą kuriančios organizacijos priėmė Lean mąstymą ir pradėjo diegti Lean pricipus produkto kokybei ir sąnaudoms, kas leido kurti papildomą vertė vartotojui. Tačiau Lean praktikos naudojamos nedaugelyje organizacijų (Poppendieck, 2011). Nuostolių pašalinimas iš programinės įrangos kūrimo proceso gali pagerinti produktą, kokybę ir kūrimo greitį, ko labiausiai ir pageidauja vartotojai.

Kad įtraukti Lean praktikas į programinės įrangos kūrimą, visų pirma reikia išskirti skirtumus tarp Lean principų ir Lean praktikų. Principai yra pagrindinės atitinkamos disciplinos idėjos ir įvaizdžiai, o praktikos yra tai, ką reikia padaryti, kad principai būtų realizuoti (Leffingwell, 2011). Programinės įrangos kūrimo organizacijos stengiasi adaptuoti Lean principus, kad pagerinti produktą, kokybę, sąnaudas, laiką ir greitį. Remiantis septyniais Lean principais, kuriuos pasiūlė Toyota, kai kurios praktikos buvo pasiūlytos ir programinės įrangos kūrimui. Šios praktikos buvo nukreiptos į tolygią ir nepertraukiamą proceso tėkmę ir nuostolių minimizavimą programinės

įrangos kūrimo procese (Poppendieck, 2011). 4 lentelėje pateikiami septyni pagrindiniai Lean principai.

4 lentelė. Septyni Lean principai (Poppendieck, 2011)

Lean principas	Aprašymas
Šlamšto eliminavimas	Darbas atliekamas nepertraukiamai. Gaminti produktą blogai ar blogą produktą.
Surinkimo į sistemą kokybė	Analizuoti klaidų procesą, nulinis pakantumas defektams, pastovios integracijos pagalba
Kurti žinias	Kreipti dėmesį į nedideles proceso plonybes ir naudoti mokslinį metodą
Atidėlioti įsipareigojimą	Iki paskutinio momento atsakingai priimti sprendimus, kurie įpareigoja organizaciją.
Pateikti kaip įmanoma greičiau	Greitis, kokybė, žema kaina žengia koja kojon. Darbo proceso valdymui naudoti masinio aptarnavimo teoriją
Gerbti žmones	Kiekvienas asmuo turi galią teikti prioritetus, imtis atsakomybės ir svarstyti išeitis, o ne klausytis nurodymų, ką ir kaip daryti
Viso optimizavimas	Koncentracija ir sub optimizavimo vengimas

A. Droste (2007) analizavo pagrindinius Toyota transporto priemonių gamybos inžinerinius principus. Minėtas autorius taip bandė perkelti priimtus Lean pricipus programinės įrangos kūrimo procesui. Atvejų tyrimai parodė, kad programinės įrangos kūrėjams nepavyko pritaikyti Lean principų ir praktikų. Tuo pačiu, organizacijos, kurios pritaikė Lean praktikas pasiekė pakankamo ir stabilaus našumo padidėjimo savo darbo procesuose (Poppendieck, Poppendieck, 2007). Kanban programinės įrangos kūrimas buvo analizuojamas jo gebėjimu pašalinti nuostolius, tačiau jis nepakankamai apsaugojo procesą nuo nuostolių susidarymo.

Nuostoliai suprantami kaip viskas, kas neskatina vertės vartotojui kūrimą (Middleton, Joyce, 2012). Kitaip tariant, viskas, kas nekuria pridėtinės vertės, yra nuostoliai. Kaip pavyzdys gali būti užduočių pakeitimas (laukimas), kuris yra nuostoliai, nes dėl prarasto laiko mažėja efektyvumas. Tradicinės septynios nuostolių rūšys, kurios gali atsirasti programinės įrangos

kūrime, pateiktos 5 lentelėje. M. Poppendieck ir T. Poppendieck (2007) modifikavo pagrindinius septynis gamybos nuostolius į nuostolius programinės įrangos kūrime.

5 lentelė. Nuostoliai gamyboje ir programinės įrangos kūrime (Poppendieck, Poppendieck, 2007)

Nuostoliai gamyboje	Nuostoliai programinės įrangos kūrime	Aprašymas
Atsargos	Dalinai atliktas darbas	Kai darbas yra procese, kuris nekuria vertės, kol nėra baigtas
Papildomas apdorojimas	Papildomi procesai	Papildomas proceso žingsnis, kuris nėra reikalingas. Pvz., dokumento, kuris nereikalingas, kūrimas
Perprodukcija	Papildomos savybės	Charakteristikos, kurios kuriamos, tačiau nedidina vertės vartotojui. Papildomos funkcijos
Transportavimas	Perdavimas	Dokumentų perdavimas sukuria pridėtines išlaidas
Laukimas	Vėlavimas/Laukimas	Vėlavimas kūrime ar kito pobūdžio prieš tai einančio proceso pabaigos laukimas
Judėjimas	Užduočių pakeitimas	Patikėto darbo atlikimo drausmės pažeidimai
Brokas	Klaidos	Problemos taisymas produkte

Tokiu būdu galima teigti, kad nuostolių pašalinimas programinės įrangos kūrimo procese leidžia padidinti šio proceso efektyvumą.

2.3.2. Agile XP

Ribinis programavimas (angl. Extreme Programming – XP) – tai programinės įrangos kūrimo metodas, skirtas pagerinti kuriamos programinės įrangos kokybę ir pagreitinti reagavimą į kintančius kliento poreikius. Kaip pažymi L. Beck ir C. Andres (2004), praktikoje buvo įrodyta šio metodo taikymo sėkmė eilėje skirtingo dydžio organizacijų. Agile XP yra sėkminga, nes šis

metodas pabrėžia vartotojo pasitenkinimą. Vietoje to, kad pateikti viską, ko vartotojas nori tam tikrai datai ateityje, šis metodas pateikia programinę įrangą pagal vartotojo pageidavimus, kai jam to reikia. Agile XP metodas leidžia programinės įrangos kūrėjams patikimai reaguoti į vartotojo poreikius net projekto gyvavimo ciklo pabaigoje.

XP numato komandinį darbą. Vadovai, vartotojai ir programuotojai yra lygūs partneriai bendroje komandoje. XP realizuoja paprastą, tačiau efektyvią aplinką, leidžiančią užtikrinti didelį komandos darbo našumą. Komanda save organizuoja ties problemos sprendimu, kad šią problemą išspręsti kaip galima efektyviau (Cao, Ramesh, 2008).

XP pagerina programinės įrangos kūrimo projektą penkiais pagrindiniais būdais: ryšiais, paprastumu, grįžtamuoju ryšiu, pagarba ir drąsa. XP programuotojai pastoviai bendrauja su savo klientais ir kolegomis. Jie palaiko kūrimo procesą paprastą ir „švarų“. Grįžtamą ryšį jie gauna kasdien kartu testuodami programinę įrangą. Jie pateikia sistemą klientui kaip galima anksčiau ir atlieka pataisymus, kai jų reikalaujama. Kiekviena nedidelė sėkmė padidina jų pagarbą kiekvieno iš komandos narių indėliui. Tai yra pagrindas, kurio pagalba XP programuotojai gali drąsiai reaguoti į besikeičiančius reikalavimus ir technologijas (Dybå, T.; Dingsoyr, 2009).

Agile XP gyvavimo ciklą sudaro penkios fazės: žvalgyba, planavimas, iteracijos pateikimui, gamyba, priežiūra ir pabaiga (Beck, Andres, 2004). Žvalgybos fazė paprastai trunka nuo kelių savaičių iki kelių mėnesių ir skiriama vartotojų reikalavimams pateikti, pagal kuriuos bus kuriamas pradinis produkto variantas. Tuo pačiu metu projekto komanda labiau susipažįsta su technologijomis, priemonėmis ir praktikomis, kurios bus naudojamos projekte.

Planavimo fazėje komanda praleidžia kelias dienas su klientu, kad galėtų išsiaiškinti prioritetinius poreikius pradiniam programinės įrangos variantui. Programuotojai įvertina savo galimybes, pagal kurias komandos lyderis numato kūrimo grafiką, kuris neviršija dviejų mėnesių.

Iteracijų pateikimui fazę sudaro kelios iteracijos, kad padaryti pirmą sistemos variantą. Kiekviena iteracija trunka nuo vienos iki keturių savaičių, o kiekvienos iteracijos pabaigoje atliekamas funkcionalumo testas. Paskutinės iteracijos pabaiga reiškia gamybos pradžią.

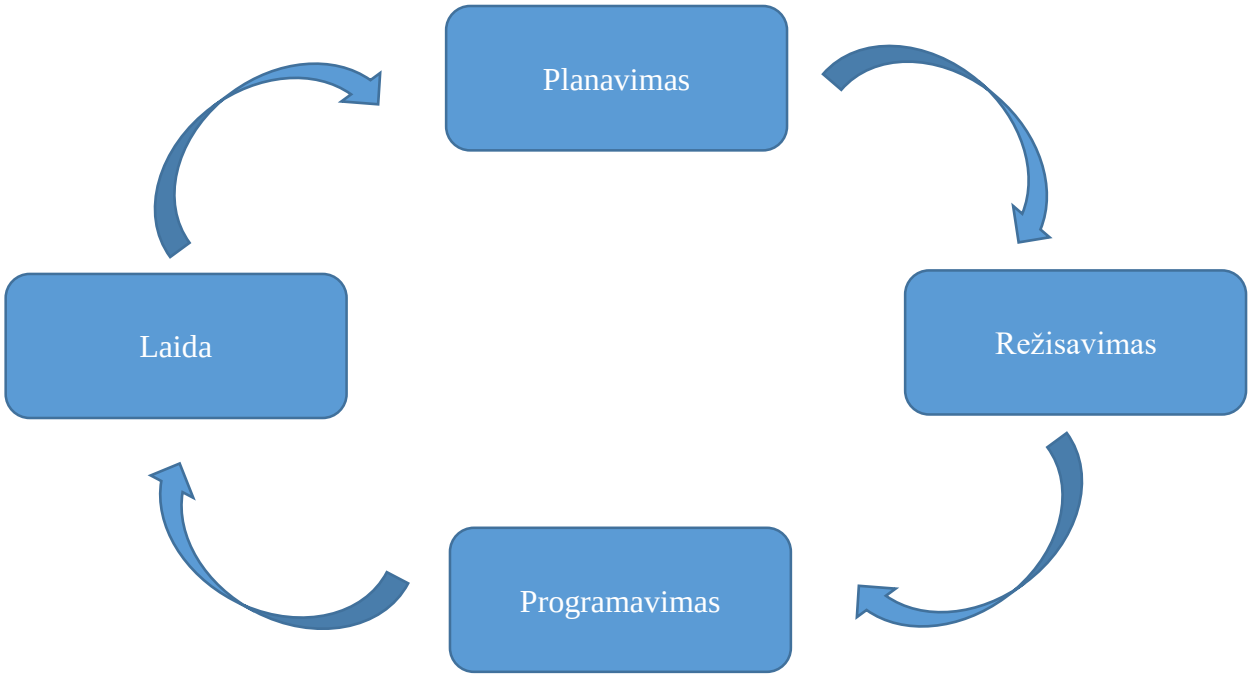
Gamybos fazėje projekto komanda atlieka papildomus našumo testus ir patikras, kad įsitikinti jog produktas atitinka vartotojo reikalavimus. Šiame etape gali būti įtraukti nauji pakeitimai ir turi būti priimtas sprendimas, kada jie turi būti įtraukti į einamą variantą. Jei jų negalima įtraukti į einamą versiją, jie turi būti dokumentuoti, kad juos būtų galima įtraukti paskesnėje versijoje. Ši fazė baigiama vartotojui perduodant einamąją versiją.

Priežiūros fazėje komanda atlieka naujas programinio produkto iteracijas, kad įtraukti pakeitimus ir funkcijas, iškilusias ankstesniame etape. Šiuos pakeitimus sudaro koreguojantys, patobulinimo ir adaptavimo pakeitimai.

Kai programinė priemonė ištobulinama ir vartotojai turi tik kelis pasiūlymus tobulinimui, prasideda pabaigos fazė. Šioje fazėje parengiama visa būtina dokumentacija. Ši fazė prasideda tada, kai daugiau nebėra pasiūlymų sistemos tobulinimui arba šie pasiūlymai yra per brangūs, kad sistemoje daryti pakeitimus.

2.3.3. Agile SCRUM

Agile SCRUM metodas apima keletą aplinkos ir techninių kintamųjų, kurie keičiasi proceso eigoje. SCRUM koncentruojasi ties tuo, kaip turi būti organizuotos komandos, kad pateiktų programinę įrangą greitai besikeičiančios aplinkos sąlygomis. Procesą sudaro keturios pagrindinės fazės: planavimas, laida, režisavimas ir programavimas (Sutherland, van Solingen, 2011)



2 pav. Agile scrum proceso pagrindinės fazės

Planavimo fazėje sukuriama projekto vizija ir numatomi lūkesčiai bei paskiriamas finansavimas. Tai atliekama planavime „prieš žaidimą“. Projektas padalinamas į iteracijas, kurios vadinamos sprintais, Kiekvieno sprinto trukmė yra 30 kalendorinių dienų. Režisavimo fazėje identifikuojami reikalavimai ir iteracijoms priskiriami prioritetai. Šios fazės pradžia – sprinto

planavimas, kurio metu formuojamas iteracijos planas. Užduotims suteikiant prioritetus, dalyvauja ir išorinės darbo grupės. Programavimo fazė apima sistemos realizavimą ir jos paruošimą pateikimui. Šiame etape sprintų darbai yra skirstomi į dienos blokus, vedančius prie kasdienio kompiliavimo. Programavimas prasideda nuo aukšto lygio projekto eskizų. Kasdien rengiamas 15 minučių trunkantis susirinkimas sprinto situacijai įvertinti. Šio susirinkimo metu komandos nariai pasirenka kokias užduotis ateityje jie turės atlikti. Sistema detalizuojama paskutinėje fazėje. Pasibaigus kiekvienam sprintui rengiamas laidos susirinkimas. Šiame susirinkime išorinei darbo grupei pristatoma sistema, kad būtų gauta atsakomoji reakcija. Tada nustatomos tolimesnės veiklos kryptys (Sutherland, van Solingen, 2011).

2.3.4. Dinaminis sistemų kūrimo metodas

Vienas iš pagrindinių aspektų, kuris skiria dinaminį sistemų kūrimo metodą nuo kitų yra tai, kad jis, visų pirma, fiksuoja laiką ir išteklius, o tada atitinkamai suderina funkcionalumą. Šis išteklių prioriteto procesas sudarytas iš penkių fazių: techninio-ekonominio pagrindimo, verslo studijos, iteracijų funkcinio modelio, iteracijų projektavimo ir kūrimo, diegimo. Paskutinės trys fazės yra pasikartojančios ir apribotos laiku: yra nustatyti terminai, kada iteracija turi baigtis laiko periode (Sani ir kt., 2013).

Techninio-ekonominio pagrindimo fazėje, įvertinamas projektas ir priimamas sprendimas ar dinaminis sistemų kūrimo metodo taikymas yra priimtinas šiam projektui. Verslo tyrimo fazėje, vertinamos pagrindinės verslo procesų charakteristikos, o taip pat naudojamos technologijos, apibrėžiama sistemos architektūra ir pradiniai plano metmenys. Architektūros apibrėžimas yra pradinė sistemos apibrėžimo versija ir gali keistis projekto eigoje. Plano prototipas apibrėžia strategijos prototipą ir vadybos būdo konfigūraciją.

Funkcinio iteracijų modelio fazės metu projektas vystomas panaudojant funkcines iteracijas, kur kiekviena iteracija apima tam tikrus pagerinimus ir papildymus, kad gauti galutinę sistemą. Šis etapas numato keturis produktus, kurie atspindi procesą: prioritetinių funkcijų sąrašą, funkcinio prototipo dokumentinę apžvalgą, nefunkcinius reikalavimus, tolimesnio vystymo rizikų analizę.

Iteracijų dizaino ir kūrimo rezultatas yra sistema, kuri patenkina minimalius reikalavimus. Iteracijos remiasi vartotojo reikalavimais. Sistemiškai realizuojant iteracijų seką, programinė įranga yra vystoma ir pateikiama vartotojui ją apžvelgti.

Diegimo fazėje sistema formaliai perduodama vartotojui ir yra suplanuojami galimi tolimesni jos tobulinimai.

2.3.5. Metodų apjungimas Agile manifestu

Agile manifestas pateikia gerą Agile metodų ketinimų apibendrinimą. Sekančios vertybės išreiškia naudojamus principus:

- asmenys ir sąveika per procesus ir priemones;
- veikiantis kodas per išsamią dokumentaciją;
- bendradarbiavimas su klientu per derybas dėl sutarties sudarymo;
- reagavimas į neplanuotus pokyčius.

Programinė įranga yra iš esmės sudėtinga dėl pastovių pokyčių. Procesai ir priemonės negali apimti visų tų pokyčių, todėl žmonės turi būti lankstūs. Vertingi žmonės per procesus suteikia daugiau kūrybiškumo sprendimams. Tai reiškia, kad net geriausias procesas negali kompensuoti individų trūkumų (Conboy ir kt., 2011).

Dokumentaciją, kuri gali būti vertinga, užima daug laiko parašyti. Tačiau ji mažiau vertinga nei veikiantis produktas. Kai kurie Agile metodai siūlo prototipo pateikimą (dinaminis sistemų kūrimo metodas), kiti skatina kurti paprastą, tačiau pilnai funkcionalų produktą kaip galima greičiau (XP).

Klientų dalyvavimas skatinamas visuose Agile metoduose. Kliento atstovai turi būti prieinami ir įsipareigoję, turintys žinių, bendradarbiaujantys, atstovaujantys ir turintys įgaliojimų. Leisti klientui greitai naudotis produktu yra bendradarbiavimo forma. Taip pat leidžiama klientui keisti savo nuomonę. Todėl vietoje sutarčių pasirašymo, kurios šiuo atveju turėtų būti keičiamos, klientai skatinami kuo aktyviau dalyvauti produkto kūrimo procese (Mafakheri, Nasiri, Mousavi, 2008).

Reagavimas į pokyčius laikomas svarbesniu nei dogmatiškas sekimas planu, nes planas yra geras tik tuo metu, kai jis yra suformuojamas. Jei kažkas pasikeičia, planas taip pat turi keistis. Tačiau pokyčiai dažniausiai būna greitesni, nei plano pakeitimas. Tačiau tai nereiškia, kad Agile metodai yra taikomi be jokio plano. Šiuo atveju bet koks planas turi būti pakankamai laisvas ir lengvai pakeičiamas. Planas turi būti rinkinys pastabų, sutašytų ant lentos, kaip tai naudojama SCRUM metode (Agarwal, Majumdar, 2012).

Šios keturios vertybės įvairiems Agile metodams suteikia lankstumo. K. Petersen ir C. Wohlin (2009) teigimu išskiriami sekantys bendri Agile metodų bruožai: bendradarbiavimas; kodo

apžvalga; mažos komandos; trumpas realizavimo grafikas; suskirstymas į laiko intervalus; pastovus testavimas.

Visi Agile metodai numato bendradarbiavimą tiek komandos viduje, tiek ir už jos ribų. Agile metodai remiasi neformaliu bendravimu, o ne didelės apimties dokumentacija, kad galėtų greitai paskleisti informaciją tarp komandos narių ir kitoms suinteresuotoms šalims. Be glaudaus bendradarbiavimo bet kuris Agile metodas pasmerktas žlugti. Tai reiškia, kad pagrindinė projekto vadovo užduotis yra užtikrinti aukšto bendradarbiavimo lygio aplinką. Projekto vadovas turi būti labiau mokytojas, nei diktatorius (Livermore, 2008).

Agile metodai taip pat skatina, jei ne reikalauja, kodo apžvalgos. Kodo apžvalga taip yra pagrindinės informacijos sklaida. Pavyzdžiui, XP metode kodo apžvalga yra nepertraukiamas procesas, nes paprastai du programuotojai naudoja tą patį kompiuterį (Cao, Ramesh, 2008).

Agile metodai taip pat skatina nedideles komandas ir nedidelį komandų skaičių projekte. Šis rodiklis svyruoja nuo vienos komandos, kurioje yra nuo trijų iki šešiolikos programuotojų, XP metode, iki šešių komandų, kurių kiekvienoje yra nuo dviejų, iki šešių narių, dinaminių sistemų kūrimo metodo atveju (Petersen, Wohlin, 2009).

Agile metodų tvarkaraščiai gali būti nuo dviejų savaičių iki šešių mėnesių trukmės (Petersen, Wohlin, 2009). Net SCRUM metodas realizavimo trukmę numato 30 dienų. Kiekvieno periodo pabaigoje funkcionalus produktas pateikiamas vartotojui, kas leidžia įvertinti produktą ir būsimus pakeitimus, kuriuos bus galima padaryti nepažeidžiant funkcionalumo vėlesnėse realizacijose.

Fiksuojant intervalus, yra užfiksuojamas galutinis terminas, bet ne atskirų intervalų trukmė. Tai yra atvirkštinis būdas „tradiciniam“, kur atskirų etapų trukmė yra fiksuota, o galutinis terminas yra kintamas. Intervalų fiksavimas padeda susikoncentruoti ties klientu ir neleidžia vėlinti produkto pateikimo jam.

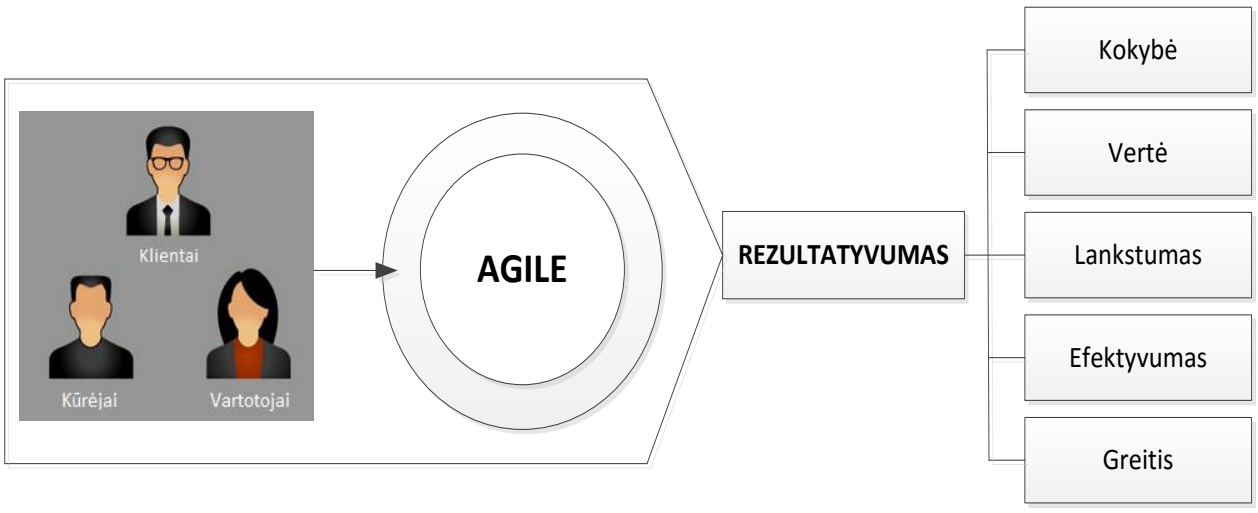
Kad padidinti pateikiamo produkto kokybę, Agile metodai didelį dėmesį sutelkia ties produkto testavimu visame projekto gyvavimo cikle. Testavimo taikymas sumažina neteisingo kodo rašymo tikimybę. Agile metodai reikalauja integruoto testavimo per visą projekto realizavimo trukmę. Visi Agile metodai taip pat numato ir bandomąjį testavimą, kurį atlieka vartotojas projekto realizavimo pabaigoje (Livermore, 2008).

Apibendrinant galima pasakyti, kad Agile metodai pateikia protingą programinės įrangos kūrimo būdą greitai besikeičiančios aplinkos sąlygomis. Tai pasiekiama Agile metodų principų pagalba. Jų teisingas panaudojimas atitinkamose aplinkybėse padeda sumažinti projekto rizikas,

padidina projekto našumą ir galutinio produkto kokybę (pavyzdžiui, mažos komandos sumažina nekokybiškos komunikacijos riziką). Be to, kai pagrindiniai Agile metodų principai derinami su kitais Agile principais, gaunam sinergija, kuri užtikrina dar glaudesnį bendradarbiavimą projekte. Agile metodai projektų vadovams siūlo alternatyvią programinės įrangos kūrimo metodologiją ir valdymą, kas užtikrina tinkamą projektų palaikymą greitai besikeičiančių reikalavimų aplinkoje. Net projektai, kurie gali atrodyti netinkami realizuoti taikant Agile metodus, Agile principus vadovai gali taikyti, kad užtikrinti projektų realizavimo efektyvumą.

2.4. Agile metodų lankstumo ir efektyvumo santykis

Agile metodo privalumas yra lankstumas, kurį suteikia Agile. Nuspręsti kaip projektas atrodys metai iki jo pabaigimo, neįvertinant pokyčio šakoje arba pokyčius užsakovo organizacijos veikloje, kurie gali padaryti poveikį projektui. Jei vartotojas pateikia savo reikalavimus pakeitimams po devynių mėnesių nuo darbų pradžios, tai gali pareikalauti pilnai pertvarkyti projektą, jei jis realizuojamas krioklio metodu. Tačiau lanksčios metodologijos adaptavimas leidžia, kad vartotojo pakeitimai būtų atlikti pagal vartotojo reikalavimus. Be to faktas, kad vartotojas gali įvertinti projekto vystymąsi ir teikti pakeitimus po kiekvienos iteracijos, vystomas gilesnis projekto suvokimas, kas reiškia, kad pakeitimai pagal vartotojo reikalavimus gali būti atliekami viso projekto gyvavimo ciklo metu.



3 pav. Agile programinių priemonių kūrimo metodo panaudojimo rezultatyvumo schema

Iš 3 pav. schemos galima pasakyti, kad Agile programinės įrangos kūrimo etapuose dalyvauja klientai, vartotojai ir kūrėjai. Pagrindinis jų bendro darbo produktas yra programinė

įranga, kurios kūrimas, panaudojant Agile metodus, pasižymi rezultatyvumu. Rezultatyvumą apsprendžia tokie Agile metodo savybės, kaip proceso kokybė, vertė, lankstumas, efektyvumas ir greitis. Daugelis iš šių savybių buvo pasiūlytos kaip atskirų Agile metodų taikymo vertinimo kriterijai.

6 lentelėje pateikiami kai kurie atitinkamų savybių parametrai, remiantis K. Bosch-Sijtsema ir kt. (2009) pateiktu programinės įrangos kūrimo rezultatyvumo apibrėžimu.

6 lentelė. Agile metodo rezultatyvumo vertinimo parametrai (Feyh, Petersen, 2013)

Savybės	Parametrai
Kokybė	Defektų skaičius Komentarai Testo efektyvumas
Vertė	Vartotojo pasitenkinimas Pateikta vertė verslui
Lankstumas	Produkto sukūrimo laikas Laukimo laikas Užduočių srautas Komandos bendradarbiavimas
Efektyvumas	Nuostoliai
Greitis	Reikalavimų pateikimo greitis Našumas

Kokybė. Defektų skaičių galima vertinti įvairiose kūrimo stadijose. Reikia pažymėti, kad atskirose iteracijose gali būti padarytas skirtingas klaidų skaičius ir toks reikšmių išsibarstymas gali formuoti pakankamai iškreiptą vaizdą (Clarke, O'Connor, 2012). Komentarai, parašyti programiniame kode sutrumpina defektų paieškos ir ištaisymo laiką. Komentarai padidina projekto rezultatyvumą ir efektyvumą, nes juose yra ta informacija, kuri talpinama dokumentacijoje. Testo efektyvumą parodo laikas, per kurį programinės įrangos defektai atrandami ir ištaisomi. Šis parametras parodo, kaip gerai vyksta procesas ir sutaupo lėšas dėl ankstyvo defektų atradimo.

Vertė. Vartotojo apklausa leidžia įvertinti ar projekto savybės buvo naudingos vartotojui. Tai labai naudinga projekto realizavimo metu, kad projektas papildomas naujomis funkcijomis ir funkcionalumas, o toks vertinimas vykdomas vartotojo dalyvavimo kūrimo procese principo rėmuose (Feyh, Petersen, 2013). Pateikta vertė verslui paprastai matuojama verte, kurią atskiros programinės įrangos funkcijos pateikia atskiriems verslo vienetams. Tai gali būti matuoja

darbuotojo atliekamo darbo trukmės sumažėjimu, darbuotojų skaičiaus dėl programinės įrangos įdiegimo sumažinimu ir pan.

Lankstumas. Produkto sukūrimo laikas nuo to momento, kai pradedamas programinės įrangos kūrimo procesas iki jo eksploatavimo pradžios. Šis laikas turi būti kaip galima trumpesnis ir neviršyti planuojamo laiko. Tai leidžia sumažinti neapibrėžtumą ir riziką projekte. Laukimo laikas yra laikas, kurį turi laukti programuotojas, kol nebus atlikta jo darbo pradžiai reikalinga užduotis. Šis laikas gali būti mažinamas teisingai paskirstant užduotis tarp projekto dalyvių. Lankstumą užtikrina dinaminis užduočių perskirstymas ir tai, kad turi būti užtikrinta tai, kad kiekvienas projekto komandos narys galėtų atlikti kelis vaidmenis (Poppendieck, Poppendieck, 2006). Kuo mažesnės eilės, tuo mažesnis laukimo laikas. Projekto realizavimo užduotys turi būti suderintos taip, kad kuo mažesnis užduočių skaičius priklausytų nuo daugelio kitų užduočių atlikimo. Žemas bendradarbiavimo lygis rodo, kad didelė projekto komandos narių bado realizuoti tą pačią užduotį.

Efektyvumas. Nuostolių eliminavimas yra vienas iš būdų padidinti programinės įrangos kūrimo rezultatyvumą. Remiantis M. Poppendieck ir T. Poppendieck (2006) yra septyni nuostolių tipai: dalinai padarytas darbas; papildomos savybės; darbuoto perkvalifikavimas; aptarnavimo perdavimas; vėlinimas; užduočių perskirstymas; defektai.

Greitis. Programinės įrangos projekto rezultatyvumą apsprendžia iškeltų reikalavimų įvykdymo greitis. Būtent tai leidžia padidinti projekto realizavimo nuspėjamumą ir įvertinti realizavimo potencialą.

Reikia pažymėti, kas rezultatyvumo savybės yra viena su kita susijusios. Lankstumas, kuris leidžia sumažinti užduočių perskirstymą, laiko nuostolius dėl užduočių atlikimo laukimo, padidina sistemos efektyvumą, nes lankstumas pats užtikrina atskirų nuostolių pašalinimą. Kokybės užtikrinimas taip pat didina ne tik vartotojo pasitenkinimą, tačiau mažina ir defektus, kas savo ruožtu didina projekto realizavimo efektyvumą. Tokiu būdu galima padaryti išvadą, kad lankstumas, kuris yra „užprogramuotas“ Agile metoduose dėl vartotojo dalyvavimo projekto realizavimo procese, didina projekto realizavimo efektyvumą.

3. TYRIMO METODOLOGIJA

Tyrimo tikslas. Ištirti ir įvertinti Agile metodų įtaką IT projektų realizavimo lankstumui ir efektyvumui ir pateikti rekomendacijas Agile metodų panaudojimui.

Tyrimo uždaviniai:

1. Ištirti Agile metodų taikymo ypatumus programinę įrangą kuriančiose įmonėse.
2. Įvertinti Agile metodų taikymo įtaką programinės įrangos kūrimo proceso lankstumui ir efektyvumui.

Siekiant įgyvendinti tyrimo uždavinius, atliktas kokybinis tyrimas, pusiau struktūruotas interviu su ekspertais – programinės įrangos kūrimo įmonių darbuotojais.

Tyrimo instrumento pagrindimas. Kokybiniam tyrimui buvo pasirinktas ekspertų struktūruoto interviu metodas. Ekspertai logiškai analizuoja kurią nors problemą, kiekybiškai vertindami ir formaliai apdorodami duomenis (Kardelis, 2007). Šis tyrimo metodas pasirinktas todėl, kad Agile metodus geriausiai išmano šioje srityje dirbantys ekspertai. Taip pat ekspertai, beveik visada būna ir to mokslo srities žinovai, geriausiai išmanantys nagrinėjamą problemą (Tidikis, 2003). Ekspertai yra sukaupę didelį kiekį racionaliai apdorotos informacijos (turi daug žinių ir patirties, gali remtis intuicija), ir todėl ekspertai gali būti kokybinės informacijos šaltiniu. Ekspertų grupės nuomonė nedaug skiriasi nuo tikrojo problemos sprendinio (Rudzikienė, 2005).

Kokybinio tyrimo atlikimui pasirinktas interviu metodas, kadangi nagrinėjamos problemos yra itin subjektyvios, netgi asmeniškos, todėl reikalingas asmeninis priėjimas prie tyrimo dalyvių. Interviu metodas leidžia priartėti prie žmonių suvokimo ir pasiekti gerą tarpusavio supratimą (R.Tidikis, 2003).

Interviu metodas gali būti dviejų tipų: struktūruotas ir nestruktūruotas (laisvasis). Laisvasis interviu yra parankus, kai reikia giliai įsigilinti į nagrinėjamą problemą ar kai mažai žinoma apie nagrinėjamą sritį. Tačiau yra rizikingas būdas, nes klausimai tyrimo pabaigoje gali radikaliai skirtis nuo tų, kurie buvo pradiniame etape, be to, leidžia interviu atlikėjui per daug įsijausti, įtakoti pokalbio eigą (Ketchen, 2005). Taip pat išskiriamas ir tarpinis variantas: pusiau struktūruotas interviu, kada yra numatomas preliminarus klausimynas – bendros gairės, tačiau pokalbis gali vykti laisvai, kai kurie klausimai atsiranda pokalbio metu, respondentas gali įtakoti interviu eigą.

Interviu yra pagrindinis kokybinių duomenų rinkimo būdas. Tai yra labai geras priartėjimo prie žmonių suvokimo, reikšmių, situacijų apibrėžimo ir realybės konstravimo (aiškinimo) būdas. Tai taip pat yra viena įtaigiausių žmonių tarpusavio supratimo priemonių (Luobikienė, 2007). Pusiau struktūruotame interviu tyrėjas yra numatęs tam tikrus klausimus ar temas, kurios bus

aptartos interviu metu. Vykstant pokalbiui tyrėjas leidžia respondentui inicijuoti naujas temas ar plėsti esamas (Girdzijauskienė, 2006). Papildomus klausimus tyrėjas užduoda esant skirtingoms situacijoms: kai interviu metu pastebi, jog numatytieji klausimai nepadengia visų tyrimui svarbių temų; siekiant surinkti daugiau ar gilesnės informacijos tuomet, kai tiriamasis nepilnai atsako į pateiktuosius klausimus; kai pastebi, jog tiriamajam nepatogu (jis nenori) atsakinėti į pateiktąjį klausimą – tuomet tyrėjas stengiasi tą pačią informaciją gauti paklausdamas kitaip ar trumpam nukreipdamas tiriamojo dėmesį į kitus, mažiau jautrius klausimus, ir sugrįždamas prie jautraus klausimo kita formuluote. Pusiau struktūruotų interviu klausimynuose taip pat beveik nenaudojami uždari klausimai su atsakymų formuluotėmis.

Atsižvelgiant į tyrimo problematiką, pasirinktas pusiau struktūruotas interviu. Jo atlikimui paruoštas preliminarus klausimynas. Klausimynas pateikiamas prieduose (1 priedas).

Tyrimo imtis. Tyrime apklausti 5 savo srities ekspertai, taip pat kai kurie ekspertai buvo apklausti pakartotinai siekiant geriau įsigilinti į analizuojamą informaciją. Taip buvo atlikti viso 9 intervių. Šie ekspertai savo darbe naudoja Agile metodus programinės įrangos kūrimo projektuose. Tai organizacijoje dirbantys vadovaujantį ar su programinės įrangos kūrimu susijusį darbą asmenys, turintys ilgą darbo įmonėje patirtį, žinantys jos veiklos ypatumus, stipriąsias ir silpnąsias puses.

Tyrimo rezultatų analizė. Kokybinis turinio analizės (*content*) metodas, kurios pagrindinis principas - respondentų pateiktų atsakymų į klausimus turinio analizė ir įvertinimas, išskiriant turinio kategorijas ir subkategorijas, jas interpretuojant ir pateikiant konkrečius atsakymus.

Tyrimo eiga. Interviu buvo tiesioginis (angl. *face-to-face*) prieš tai pateikiant informantams pusiau struktūruoto interviu klausimyną, kuris buvo nusiųstas elektroniniu paštu susipažinti. Interviu buvo vedamas kasdienine kalba, nepateikiant respondentui nesuprantamų klausimų. Esant neaiškumams, klausimai buvo paaiškinami plačiau.

Tyrimo etika. Respondentams buvo pateikiama informacija iš kokios aukštosios mokyklos yra magistrantas, kokio kurso ir mokymo programos. Taip pat nurodomas tyrimo objektas ir tikslas. Interviu anoniminis. Tyrimo metu buvo laikomasis etikos principų:

- laisvanoriškumo – informantas dalyvauja savanoriškai, niekieno neverčiamas;
- konfidencialumo – užtikrinamas apie tiriamąjį ir iš jo gautos informacijos konfidencialumas. Niekas be tiriamojo sutikimo negali naudotis jo pateikta informacija, išskyrus tyrėją;

- anonimiškumo – užtikrinamas tiriamojo anonimiškumas viso tyrimo metu ir po jo;
- laikomasi etikos normų – vengiama šališkumo.

Norint užtikrinti informantų konfidencialumą gauti sutikimai niekur neviešinti asmens duomenų ir darbe nepateikiami. Visi duomenys saugomi tyrėjo asmeniniame archyve.

4. AGILE METODŲ LANKSTUMO IR EFEKTYVUMO UŽTIKRINIMO TYRIMAS

4.1. Atlikto interviu tyrimo rezultatai ir analizė

Pirmu klausimų moduliu buvo siekta išsiaiškinti apie Agile metodų taikymo programinės įrangos kūrimo įmonėse, bei šių metodų taikymo pasirinkimo motyvus.

7 lentelė. Agile metodų taikymo paplitimo ir naudojimo motyvų vertinimas

Klausimas	Ekspertų atsakymai
1. Jei žinote, kada Agile metodai pradėti taikyti jūsų įmonėje? Gal žinote, kodėl įmonė ėmė taikyti Agile?	„galutinis produktas greičiau pateikiamas rinkai“, „geresnė sąveika su vartotoju“, „faktiškai nereikia daryti pakeitimų po to, kai klientas pradeda dirbti su produktu“
2. Kokį Agile metodą Jūs naudoja jūsų įmonė: Agile-Lean, Agile XP, Agile SCRUM, Dinaminis sistemų kūrimo metodą ar kokį kitą? Jei žinote, kodėl buvo pasirinktas šis?	„leidžia keisti reikalavimus produktui kiekvienu momentu“, „suteikia disciplinos“, „leidžia įtraukti vartotoją į kūrimo procesą“; „leidžia greitai gauti pirmą programos versiją“
3. Jei žinote, kokiai Jūsų vykdomų projektų daliai naudojate Agile metodus? Kokio tipo programinės įrangos kūrimo projektams naudojate Agile metodus? (pavyzdžiui, Interneto Duomenų apdorojimo, Vartotojo sąsajos, Atskira programa, Verslo valdymo sistema, Realaus laiko/Valdymo, Sistemos (OS ir pan.)	„verslo valdymo sistemoms“, „visiems projektams“

Ekspertų atsakymų apie Agile metodų taikymo trukmę ir kokius Agile metodus taiko kiekviena įmonė suvestinė pateikta 2 lentelėje.

8 lentelė. Agile metodų naudojimas respondentų atstovaujamose organizacijose

	UAB „X1“	UAB „X2“	UAB „X3“	UAB „X4“
Naudojimo pradžia	2009	2011	2010	2011
Agile Scrum	+	+	+	+
Agile XP		+		
Dinaminis sistemų kūrimo metodas				
Agile Lean				

Visų pirma reikia pažymėti, kad visos respondentais atstovaujamos įmonės Agile metodus savo veikloje naudoja jau daugiau nei penkis metus, o tai leidžia padaryti išvadą, kad jos turi

pakankamai patirties, kad galėtų vertinti Agile metodų naudojimą. Taip pat pažymėtina, kad visose apklaustose įmonėse naudojamas SCRUM metodas ir tik vienoje dar papildomai naudojamas Agile XP metodas. Todėl galima padaryti prielaidą, kad SCRUM metodas yra populiariausias tarp programinę įrangą gaminančių įmonių. Taip pat kaip vieną iš tokio metodo privalumų ekspertas E3 nurodė tai, kad panaudojus tokį metodą, „*vartotojui greitai galima pateikti veikiantį produkto variantą, o tai padidina vartotojo pasitikėjimą gamintoju*“. Tačiau ekspertas pastebėjo, kad „*po tokios demonstracijos vartotojo reikalavimai keičiasi arba jų pradeda daugėti*“

Paprašyti paaiškinti naudojamo Agile metodo pasirinkimo motyvus, respondentai pateikė visą eilę tokių motyvų. Visų pirma reikia pažymėti, kad ekspertai pažymėjo, kad apie tokius metodus jie išgirdo „*iš kitų įmonių*“ ir tik vienas pažymėjo, kad susidomėjimą šiuo metodu sukėlė „*pasakojimas apie jį viename iš seminarų*“. Dažniausiai ekspertai tokį motyvą, kad toks metodas leidžia įtraukti vartotoją į produkto kūrimo procesą. Taip pat kaip svarbų motyvą, paskatinusi naudoti Agile metodus, ekspertai pažymėjo „*galimybę greičiau pateikti produktą rinkai*“. O tai savo ruožtu reiškia greitesnį pajamų gavimą, galimybę vykdyti daugiau projektų, bei lemia įmonės veiklos ekonominio efektyvumo augimą.

Ekspertas E1 pažymėjo, kad jo atstovaujamoje įmonėje Agile metodas taikomas „*visiems projektams, nepriklausomai nuo jų pobūdžio*“. Ekspertas E2 pažymėjo, kad sunku pasakyti, kokiems konkretiems projektų tipams naudojamas Agile metodas, tačiau „*stengiamės kuo daugiau naudoti šį metodą*“. Ekspertas E3 pažymėjo, kad Agile geriausiai yra naudoti tokiems projektams, kuriuose „*dažnai keičiasi užsakovo reikalavimai*“. Todėl apibendrinant ekspertų atsakymus galima pasakyti, kad šis metodas geriausiai naudotinas tokiuose projektuose, kuriuose reikalinga greitai pateikti veikiančio produkto variantą arba kur vartotojo reikalavimai dažnai keičiasi. Tarp tokių projektų paminėtini verslo valdymo sistemos, duomenų apdorojimo sistemos, veikiančios sistemos papildymas nauju moduliu.

Sekančiu klausimų moduliu siekta išsiaiškinti ekspertų nuomonę apie Agile projekto įgyvendinimo procesą.

9 lentelė. Agile metodo įgyvendinimo proceso vertinimas

Klausimas	Ekspertų atsakymai
Gal galite kiek įmanoma detaliau papasakoti kaip vyksta projekto įgyvendinimas taikant Agile, remiantis konkrečiu projektu?	„labai sunku detaliai nupasakoti visą procesą“, „SCRUM yra principų rinkinys, o jo panaudojimas gali skirtis skirtinguose projektuose“
Kokias programines priemones naudojote programinių priemonių kūrimo projektuose, taikant Agile metodą?	„Jira“, „TRAC“
Kokios organizacinės priemonės buvo taikomos naudojant Agile metodą?	„kiekvienam projektui sudaroma komanda“, „visi dalyviai padalinami į dvi grupes“, „suplanuojami darbuotojai, kurie gali pakeisti komandos narį ar papildyti komandą“
Su kokiomis problemomis susidūrėte naudodami Agile programinės kūrimo metodą?	„Agile metodą buvo bandyta diegti dalinai“, „komandos sudarymo sunkumai“

Į klausimą apie naudojamas programines priemones trys ekspertai paminėjo Jira, o du ekspertai – Trac. Reikia pažymėti, kad Trac programinė priemonė yra nemokama, o Jira – mokama. Ekspertai pažymėjo, kad Jira yra „*lanksti, lengvai konfigūruojama, paprasta naudotis*“. Taip pat tokių programinių priemonių naudojimas leidžia planuoti ateities darbus, nes Jira panaudojimas leidžia “*sekti darbų vykdymą, kontroliuoti jų eiliškumą*“. Ekspertai pripažįsta, kad programinių priemonių panaudojimas leidžia vykdyti kontrolę ir padidinti projekto vykdymo efektyvumą. Ekspertas E5 paminėjo, kad tokių programinių priemonių panaudojimas leidžia „padidinti projekto vykdymo skaidrumą“, o taip pat leidžia matyti „*kas kokias užduotis vykdo ir kokioje stadijoje yra jo vykdoma užduotis*“.

Kaip paminėjo ekspertai E1 ir E2, remdamiesi Agile principais, visus projekto dalyvius padalina į dvi grupes: „*specialistai, kurie dalinai įtraukiami į projektą ir betarpiški projekto vykdytojai*“. Viena iš grupių, tai vartotojai, ekspertai-konsultantai ir suinteresuotos šalys (klientai ir pardavėjai). Kitą grupę sudaro projekto vadovas, kuris vadovauja susirinkimams, sprendžia prieštaravimus ir problemas, saugo komandą nuo pašalinių dirgiklių. Projekto komanda, kurią sudaro įvairių sričių specialistai: programuotojai, testuotojai, analitikai. Taip pat į šią grupę įtraukiamas produkto savininkas, kuris atstovauja vartotojų ir kitų suinteresuotų šalių interesus.

Taip pat ekspertas E4 pažymėjo, kad kiekvienam projektui yra suplanuojami darbuotojai, kurie reikalui esant (susirgus komandos nariui, ar matant, kad projektui pabaigti laiku teikia papildomų specialistų) pakeičia komandos narį ar įsitraukia į komandos darbą.

Vienas iš sunkumų, kurį paminėjo ekspertas E3, diegiant Agile metodą, buvo tai, jog Agile metodo taikymą pradėjo nuo vienos komandos, tačiau laukiamo efekto nesulaukė. Tačiau reikia pažymėti, kad Agile metodo negalima diegti tik vienoje komandoje. Visa organizacija turi būti lanksti. Kaip pažymėjo ekspertas, „*supratome vartotojo dalyvavimo svarbą, kai pakvietėme suformuluoti užduotis kartu su komanda*“. Tokiu būdu labai svarbu, kad būtų diegiami pilni Agile metodai ir projekto realizavime dalyvautų vartotojas ar jo interesų atstovas. Kita problema, kurią minėjo ekspertai – „*sunkumai su savo srities profesionalų pritraukimu*“, o taip pat „*nepasiruošimas tam, kad žmonės suserga, o jų nėra kam pakeisti*“. Kompetentingos Agile komandos problema gali būti sprendžiam tik vienu būdu: reikalinga diegiant šiuolaikinius procesus, įmonę daryti kuo patrauklesne darbuotojams ir pritraukti kuo aukštesnės kvalifikacijos personalą. Sprendimas apie naujo darbuotojo priėmimą turi būti sprendžiamas kolegialiai, kas darbuotojams suteiks daugiau atsakomybės ir autonomijos. Pati Agile metodika atmeta viršininkų būtinumą. Būdama susitelkusi, komanda, kai koks jos narys suserga ar išeina gali mobilizuoti visas savo pastangas būtiniems uždaviniams spręsti. Todėl įmonės veiklos efektyvumui padidinti reikalingos ir atitinkamos priemonės, kurias panaudojus, padidėtų darbuotojų įsitraukimas į darbą ir didėtų įmonės patrauklumas.

Kitu klausimų moduliu siekta išsiaiškinti ekspertų nuomonę apie Agile metodų taikymo privalumus ir trūkumus.

10 lentelė. Agile metodų taikymo privalumai ir trūkumai

Klausimas	Ekspertų atsakymai
Kokias teigiamas pasekmes teikia dažniausiai Jūsų įmonėje naudojamo Agile metodo taikymas?	„kūrimas iteracijomis“, „vartotojas įtrauktas į kūrimo procesą“, „greitai sukuriama bandomoji produkto versija testavimui“, „lengvai atliekami koregavimai darbo procese“
Kokias neigiamas pasekmes teikia dažniausiai Jūsų įmonėje naudojamo Agile metodo taikymas, jei tokių yra?	„gali būti, kad galutinis produktas bus žemos kokybės“, „rizika niekada nepabaigti projekto“, „gali būti problemų su produkto plėtra“

Visų pirma galima pastebėti, kad ekspertai atrado daugiau Agile metodo privalumų nei trūkumų. E1 ir E3 ekspertai kaip pagrindinį Agile metodo privalumą paminėjo, kad jo pagalba „*greitai sukuriama bandomoji produkto versija testavimui*“. Tai leidžia patikrinti produkto funkcionalumą, o vartotojo dalyvavimas leidžia greitai įvertinti produkto atitikimą jo reikalavimams ir vartotojui, reikalui esant, suformuluoti naujus uždavinius, kuriuo lengvai realizuoja komanda. Ekspertas E5, kaip privalumą pažymėdamas „*kūrimo iteracijomis galimybę*“, pakomentavo, jog toks kūrimas leidžia panaudoti efektyvumo metrikas, kai kiekvienoje iteracijoje skaičiuojami atlikti uždaviniai, o kadangi yra įvedama sprinto sąvoka, kurių trukmė yra pastovi, galima paskaičiuoti programinio projekto kūrimo efektyvumą. Taip pat kaip privalumą daugelis ekspertų pažymėjo, kad „*vartotojas įtrauktas į kūrimo procesą*“. Tai leidžia greitai ir tiksli išsiaiškinti vartotojo poreikius, o vartotojas pats gali formuluoti uždavinius kūrimo komandai.

Kai trūkumą ekspertas E3 išreiškė nuogąstavimą, kad „*gali būti, kad galutinis produktas bus žemos kokybės*“. Tai gali lemti tai, jos greito bandomosios versijos sukūrimo metu gali būti praleistos kokios nors esminės savybės, nes visas dėmesys gali būti sutelktas į produkto funkcionalumą. Taip pat gali būti rizika, kad projektas niekada nebus baigtas, nes po bandomosios versijos gali vis didėti vartotojo reikalavimai, kurie gali versti produkto kūrimo komandą keisti jau anksčiau padarytus ir išbandytus sprendimus.

Sekantys klausimai yra skirti išsiaiškinti kaip pasiekiamas lankstumas ir efektyvumas atskiruose programinės įrangos kūrimo etapuose.

11 lentelė. Lankstumo ir efektyvumo vertinimas paruošimo etape

Teorinė kategorija	Pirmos eilės kategorija	Nagrinėjamas rezultatas	Citata
Lankstumas ir efektyvumas	Konkretaus projekto planavimas	Kokybė	„su vartotoju aptariamas kuriamos programinės įrangos funkcionalumas“; „išsiaiškinami vartotojo lūkesčiai dėl programinės įrangos veikimo“
		Vertė	„su vartotoju suderinama projekto kaina“; „nustatoma bazinė kaina ir numatomi priedai už funkcionalumą“;
		Greitis	„nustatomas vėliausias projekto pabaigos laikas“; „vartotojas visada nori kuo greičiau, todėl diskusijose ieškoma kompromisinio varianto“

	Projekto planavimas pagal analogiją	Kokybė	„dažnai vartotojas nelabai įsivaizduoja, kaip jo užsakoma programinė įranga turi veikti“; „bandoma ieškoti konkrečių vartotojo atstovų, kurie žinotų konkrečias sritis, kuriose bus naudojama kuriama programinė įranga“
		Vertė	„vertė dažniausiai nustatoma pagal jau darytą panašią programinę įrangą“; „vartotojui pateikiama kaina, tačiau projekte ji dažniausiai išauga“
		Greitis	„paprastai be vartotojo labai sunku nustatyti projekto realizavimo trukmę“, „bandoma remtis analogišku projektu“

Kalbant apie lankstumo ir efektyvumo užtikrinimą Agile metodo taikymui programinės įrangos kūrime, išsiaiškinta, kad didžiausią reikšmę tokiam užtikrinimui turi vartotojo dalyvavimas šiame etape. Šiuo atveju galimi du variantai: kai vartotojas aktyviai dalyvauja pasiruošimo ir planavimo procese ir kitas variantas, kai vartotojo dalyvavimas yra minimalus. Pirmu atveju tikslinga vartotojui pateikti reikalavimus, suformuluoti jo funkcionalumo lūkesčius, terminų ir kainos atžvilgiu. Tačiau vartotojo ir programinės įrangos gamintojų kainos skiriasi. Taip yra dėl tos priežasties, kad vartotojas yra žemiausiame apibrėžtame taške ir dar nėra apsisprendęs dėl reikalavimų pageidaujamai programinei įrangai. Pasinaudojant bendradarbiavimu su programinės įrangos gamintoju, išanalizavęs analogiškos programinės įrangos funkcionalumus ir sukūrimo galimybes, vartotojas apsisprendžia dėl funkcionalumo, suderinami terminai ir kaina.

Kitu atveju, be vartotojo bendradarbiavimo, programinės įrangos kūrimo įmonė gali suformuluoti funkcionalumo reikalavimus pagal jau analogiškus atliktus projektus, padidinti kainą, kad galima būtų apsidrausti nuo rizikų, susijusių su visišku neapibrėžtumu pradiniuose projekto realizavimo etapuose. Tokiu būdu prarandamas efektyvumas, nes vartotojui projekto realizavimas pagal šį modelį kainuos gerokai brangiau nei bendradarbiavimo atveju.

Tokiu būdu galima pasakyti, kad Agile lankstumą ir efektyvumą projekto planavimo ir paruošimo etape užtikrina bendradarbiavimas su vartotoju. Lankstumas užtikrinamas tuo, kad bendradarbiaujant su vartotoju gaunami reikalavimai funkcionalumui (kokybei), vertei (projekto kainai) ir greičiui (programinės įrangos sukūrimo laikui). Visa tai užtikrina projekto realizavimo

efektyvumą: projektas bus pateiktas su reikiamu funkcionalumu, tinkamais terminais ir už prieinamą kainą.

12 lentelė. Lankstumo ir efektyvumo pasiekimas Agile metodo taikymo procese

Teorinė kategorija	Pirmos eilės kategorija	Citata
Lankstumas ir efektyvumas	Su Agile metodu siejamos darbo priemonės	„Trac“, „Jira“,
	Su Agile siejamos organizacinės priemonės	„išskirčiau tai, kad numatomi du pagrindiniai asmenys: vienas atsakingas už ryšius su vartotojų, o kitas – už darbuotojų mokymus ir motyvavimą“; „paprastai komandoje yra 7 žmonės, nes didesnis skaičius mažina komunikacijos efektyvumą, o mažesnis didina rizikas dėl galimo kvalifikacijos trūkumo“
	Agile metodo taikymo problemos	„mes naudojame SCRUM metodą, o jame yra numatyta keletas griežtų taisyklių, kas kertasi su vartotojo dalyvavimu“; „Agile metodai orientuoti į saviorganizavimą, o tai mažina išlaidas komandos koordinavimui, tačiau padidina išlaidas personalo atrankai, mokymams ir motyvavimui“

Trys iš apklaustų organizacijų naudoja Jira programinę įrangą. Ekspertų pastebėjimų ji pakankamai padidina projekto realizavimo efektyvumą, nes kiekvienas komandos narys su Jira gali valdyti jam priskiriamas užduotis, stebėti projekto vykdymo eigą, kurti užduotis kitam projekto komandos nariui. Būtent toks užduočių kūrimas užtikrina lankstumą, nes kiekvienas komandos narys gali paskirti užduotis kitam nariui. Track leidžia peržiūrėti pateiktas užduotis, užduočių koregavimus, papildymus, užbaigimus. Taip pat parodo atsakingus užduočių asmenis ir tuos, kuriems yra priskiriamos užduotys atlikti. Todėl galima padaryti prielaidą, kad programinės įrangos kūrimo Agile metodu efektyvumas tam tikra dalimi priklauso ir nuo naudojamos programinės įrangos.

Iš ekspertų atsakymų galima padaryti išvadą, kad kalbant apie organizacines priemones, Agile metodų efektyvumą užtikrina du pagrindiniai vaidmenys – tai *product owner,* kuris yra jungianti grandis tarp komandos ir vartotojo. Pagrindinis *product owner* uždavinys – maksimaliai padidinti kuriamos programinės priemonės ir komandos darbo vertę. Kitas svarbus vaidmuo yra *scrum master*, kuris padeda komandai maksimizuoti jos efektyvumą, pašalinant kliūtis, teikiant

pagalbą, mokymus ir motyvaciją. Lankstumą savo ruožtu užtikrina komanda, kuri yra save organizuojanti, nes jai niekas negali nurodyti kaip ir kokiu būdu reikia didinti, kad pasiekti reikiamus rezultatus. Taip pat lankstumą užtikrina komandos multifunkcionalumas, nes kiekvienas komandos narys turi turėti tuos gebėjimus, kurie padėtų pasiekti galutinį tikslą. Narių tarpusavio pakeičiamumas leidžia šiuos gebėjimus panaudoti ten, kur jų tuo metu labiausiai reikia. Savo ruožtu komandos darbo efektyvumą užtikrina ir atsakomybės paskirstymas, nes už atliekamą darbą atsako visa komanda, o ne atskiri jos nariai.

Kaip pagrindines problemas, trukdančias lankstumo ir efektyvumo realizavimui, ekspertai įvardina tai, kad Agile metodai turi nors ir nedaug, tačiau griežtas tam tikras taisykles. Tai konfliktuoja su orientacija į klientą, nes vartotojui nėra svarbios kūrimo komandos vidaus taisyklės, ypač jei jos riboja vartotoją. Pavyzdžiui, klientas gali pats pakeisti užduotis, net jei kūrimo komanda tam priešinasi. Savo ruožtu Agile nenumato kokio nors komunikacijų ir reagavimo į rizikas plano. O tai apsunkina ar daro neįmanomą formalų (teisinį ar administracinį) pasipriešinimą Agile taisyklių pažeidimams. Kita problema yra save organizuojanti komanda. Tam tikrais atvejais mažina jos veiklos efektyvumą. Nors ir mažėja išlaidos komandos koordinacijai, nes jos viduje patys komandos nariai skirstosi vaidmenis ir užduotis, tačiau padidėja išlaidos tokių komandos narių paieškai, jų apmokymui ir motyvavimui.

Apibendrinant galima pasakyti, kad Agile darbo ir organizacinės priemonės užtikrina tiek lankstumą, tiek ir efektyvumą. Agile organizacinė struktūra užtikrina lankstumą komandos viduje, o tinkama pagalba ir ryšio su vartotoju palaikymas užtikrina efektyvumą produkto kokybės, pateikimo ir sukuriamos vertės atžvilgiu. Tačiau komandos savęs organizavimas mažina efektyvumą. Efektyvumo padidėjimą galima padidinti komandos pastovumu jos narių požiūriu ir tada nereikalingos išlaidos naujų narių paieškai, jų mokymui ir motyvavimui.

13 lentelė. Agile metodų privalumai ir trūkumai lankstumo ir efektyvumo užtikrinimo kontekste

Teorinė kategorija	Pirmos eilės kategorija	Citata
Lankstumas ir efektyvumas	Agile metodo taikymo privalumai	„gerai galima matyti žmonių darbo greitį“; „projektas stebimas“; „visi žino kas ką turi veikti“; „problemos greičiau pastebimos ir pašalinamos“; „greitesnis rezultatas“

	Agile metodo taikymo trūkumai	„daug valdymo ir koordinavimo darbo“; „per didelė kontrolė“; „iš esmės Agile taikymas nereiškia visiško klasikinių metodų atsisakymo“

Kaip pagrindinį Agile metodų taikymo privalumą ekspertai pažymėjo skaidrumą, t.y. galimybę aiškia matyti kas ką šiuo metu daro ir tokiu būdu greitai galima pastebėti klaidas ir jas pašalinti, o tai užtikrina lankstumas. Todėl šiuo atveju lankstumas, kaip Agile metodų privalumas, užtikrina kokybės ir greičio, kurie yra efektyvumo kriterijai, užtikrinimą.

Taip pat, ekspertų nuomone, Agile metodai turi ir trūkumų. Visų pirma tai komanda. Tuo metu, kai tradicinis valdymas sukoncentruotas ties griežto veiksmų atlikimo plano laikymosi, o vėliau ties rezultatu, naudojant Agile metodus didžiausias dėmesys skiriamas klientui ir rezultatui. Agile nereiškia visiškos laisvės suteikimo. Ji reiškia saviorganizavimą su tam tikrais apribojimais, kuriuos nustato vadovas. Agile metodas bet kuriuo atveju reiškia, kad reikalingas geras išankstinis planavimas.

14 lentelė. Agile lankstumo ir efektyvumo užtikrinimas

Teorinė kategorija	Pirmos eilės kategorija	Citata
Lankstumas ir efektyvumas	Lankstumo ir efektyvumo santykis	„vėlinimo efektas“; „mažiau klaidų“; „geresnė kokybė“; „greitas rezultatas“

Efektyvumo skaičiavimui galima pasinaudoti „vėlinimo efektu“. Labai retas kuris projektas vykdomas aplenkiant grafiką. Tegul projektas apmokamas etapais ir žinoma, kad pirmam etapui bus išleista N eurų atlyginimams, biuro išlaikymui ir kitoms išlaidoms Iš užsakovo gaunama M eurų. Tada tikėtinas pelnas yra M-N eurų, o pirmo etapo trukmė tegul būna K mėnesių.

Klasikiniu programinės įrangos kūrimo atveju suprantama, kad projektas vėluos ar ne, projekto realizavimo viduryje. Paprastai jei nespėjama, iš karto siūlomos idėjos „dirbti per išeigines“, „vėliau užduotys bus paprastesnės“ ir pan. Rezultate, po 0,7K suprantama, kad tikrai bus nespėta ir visą pelną (M-N) „suvalgys“ vėlinimas. Tada pasirinkimas yra tarp varianto dirbti į nuostolius ar nurašyti 0,7K eurų. 0,7 gaunamas sekančiai: po 50 proc. laiko atsirado pojūtis, kad nebus spėta, o dar po 20 proc. tai pasidarė taip akivaizdu, kad to jau negalima ignoruoti.

Tegul etapas – tai keturi mėnesiai ir keturiems darbuotojams mokama po 1500 eurų. Dar tiek pat išleidžiama nuomai, mokesčiams ir pan. Tada N=4*(4+1)*1500. Tokiu būdu bendra

atlyginimų suma sudarys 30000 eurų. Iš užsakovo gaunama (M) 50000 eurų. Tada trečio mėnesio pabaigoje, kai jau nebebus galima užmerkti akių į atsilikimą, matysis, kad darbo dar liko dviem mėnesiams ir bus pasirinkimas: nurašyti ar ne 30000*0,7, t.y. 21000 eurų.

Pagal Agile SCRUM metodiką, o taip pat savaitines iteracijas, ženklią ir pakankamai tikslią prognozę po šešių iteracijų. Tai reiškia, kad antro mėnesio pradžioje-vidutyje bus žinoma, bus spėta laiku at ne: N=30000*1,2/4=9000 eurų. Reikia pažymėti, kad SCRUM naudoja *Velocity* praktiką. Ji leidžia paskaičiuoti komandos greitį pagal užbaigtų darbo dalių skaičių per laiko tarpą.

Tokiu būdu, dirbant pagal klasikinę schemą, rizikuojama 21000 eurų, o dirbant pagal SCRUM – 9000 eurų.

15 lentelė. Veiksnių, užtikrinančių lankstumą ir efektyvumą, tyrimas

Teorinė kategorija	Pirmos eilės kategorija	Nagrinėjamas rezultatas	Citata
Lankstumas ir efektyvumas	Prisidedančios prie projekto efektyvumo	Įtraukimas	„taikant tradicinius metodus, atsakomybė paskirstoma tarp dalyvių“; „pagal Agile visi vienodai įtraukiami į kūrimą ir vienodai atsako už viską“; „susidariusios problemos – tai visų problemos“
		Inovatyvūs metodai	„naudojant Agile metodą, galima naudoti kad ir Lean koncepciją, kuri mažina nuostolius ir didina efektyvumą“
	Prisidedančios prie projekto lankstumo	Įtraukimas	„kad užtikrinti Agile lankstumą, turi būti sukurta infrastruktūra, lanksčiai reaguojanti į bet kokius pokyčius“; „reikalinga kuo daugiau analizuoti įvairios informacijos ir ja dalintis“
		Inovatyvūs metodai	„paminėčiau save organizuojančią komandą, nes ankstesniuose metoduose buvo taikoma griežta struktūra, kad savaime atmeta lankstumą“
	Prisidedančios prie projekto lankstumo ir efektyvumo	Lyderystė	„vadovas privalo kasdien atnešti kažką naujo organizacijai“
		Integracija	„kad suderinti lankstumą ir efektyvumą, reikia lanksčios programuotojų ir produkto administratorių sąveikos“

		Iteracijų taikymas	„užtenka pasakyti, kad pradedam eilinį savaitės etapą, apibrėžti tikslą ir komanda pati pasikirsto užduotis“; fiksuotas iteracijų laikas moko programuotojus įvertinti terminus“

Agile efektyvumas pasiekiamas ir tuo, kad kiekvienas projekto dalyvis įtraukiamas į projekto vykdymą visuose etapuose. Kiekvienas gali teikti savo pasiūlymus, į kuriuos bus atsižvelgta. Naudojant tradicinius projektavimo metodus, atsakomybės yra paskirstoma tarp atskirų padalinių ar grupių, kurios dažnai konfliktuoja tarpusavyje. Kaip pažymėjo informantas „niekam nėra paslaptis, kad eksploatavimo skyriaus darbuotojai, testuotojai ir programuotojai dažniausiai konfliktuoja tarpusavyje. Jei projektas nesiseka, kiekvienas kaltina kitą, nors dažniausiai, kaip taisyklė, kalti yra visi“. Agile metodas numato visų programinės įrangos produkto kūrimo proceso dalyvių įtraukimą, paliekant jiems įprastas kompetencijas. Toks būdas leidžia suprasti, kad jie visi dirba vienam ir tam pačiam galutiniam tikslui – sukurti kokybišką produktą savo klientui. Taip pat toks visų dalyvių įtraukimas keičia programinės įrangos kūrimo įmonės verslo kultūros pokyčius. Taikant Agile metodą, kiekvienas projekto dalyvis daro viską, todėl susiformuoja draugiškas kolektyvas, efektyviai veikiantis rinkoje. Efektyvumo požiūriu tai ideali organizacinė struktūra. Paprastai taikant Agile metodą suformuojamos komandos, kurios dėl bendro tikslo tampa emociškai kolektyvios ir kurios dirba galutiniam tikslui – sukurti veikiantį kokybišką programinį produktą. Todėl problemos atsirandančios bet kuriame etape – yra visų komandos narių ir kitų asmenų, įtrauktų į kūrimo procesą, problemos ir visi, kurie sugeba jas išspręsti yra įtraukiami į šį procesą.

Kad padidinti Agile metodo taikymo efektyvumą, tikslinga naudoti ir kitas koncepcijas, pavyzdžiui taupios gamybos Lean koncepciją. Tokio panaudojimo tikslas – sumažinti nereikalingų veiksmų kiekį, kurie produktui nesuteikia pridėtinės vertės arba ją mažina. Tai gali būti pasikartojančio kodo dalies rašymas vietoje to, kad panaudoti jau parašytą, dubliuojančio funkcionalumo realizavimas. Tokiu būdu mažinami nuostoliai programinės įrangos kūrime, o tai didina pačio kūrimo proceso efektyvumą.

Kuriant programinę įrangą naudojantis klasikiniais metodais, reikalingas ilgalaikis planavimas. Tačiau jei kas nors pasikeičia, pavyzdžiui, prireikus daugiau serverių at kitų komponentų, galimas toks scenarijus, kad projekto realizavimas sustos: reikės ieškoti naujų serverių, juos pirkti, derinti. Tokiu būdu Agile tampa ne tik paprasta naujos programinės įrangos kūrimo metodologija, tačiau visos organizacijos vystymosi lankstaus planavimo sistema. Turi būti

sukurta tokia infrastruktūra, kuri taip pat lanksčiai reaguotų į klientų poreikius ir reikalavimus, kurie keičiasi programinės įrangos kūrimo procese ir jo eksploatavime.

Lankstumas užtikrinamas ir per komandos saviorganizavimą. Tai leidžia atsisakyti bereikalingos vadybinės struktūros, nes egzistuoja tik vartotojo atstovas, kuris atstovauja verslo interesus. Be to, nebūtina tikrinti kiekvieną dalyvį: komanda pati pasiskirsto užduotis ir vidinę atsakomybę, o taip bus garantuotas darbo našumas ir produkto kokybė. Pati saviorganizacija yra komandos motyvavimas.

Lankstumas ir efektyvumas užtikrinamas ir lyderystės bei atsakomybės pagrindu. Visų pirma pats vadovas kasdien turi pateikti organizacijai kažką naujo ir ši inovacijų siekis tampa bendros organizacijos kultūros pagrindu. Inovacinė kultūra formuojasi komandos formavimo principuose, diskusijų, kurios vyksta visuose susirinkimuose, temose, tikslų ir uždavinių nustatyme, organizacijos misijoje ir vizijoje. Tokiu būdu galima teigti, kad suformuota inovatyvi organizacijos kultūra Agile metodų naudojimo pagrindu skatina per lankstumą pasiekti efektyvumą.

Inovatyvių koncepcijų panaudojimas leidžia lankstumo pagalba užtikrinti efektyvumą. Agile metodo taikymas turi užtikrinti nepertraukiamą produkto pateikimą. Greitų iteracijų rėmuose iškyla programuotojų ir administratorių, kurie eksploatuoja programinę įrangą lanksčios integracijos problema. Tam gali būti panaudojama DevOps metodologija, kuri užtikrina aktyvų programuotojų ir administratorių bendradarbiavimą. Remiantis šia metodologiją sukuriamas vieningas programinės priemonės kūrimo, eksploatavimo ir pateikimo ciklas, kurio pagalba lengvai atnaujinama programinė priemonė. Tai derinasi su Agile metodologija, kai vartotojui pateikiamas pradinis variantas, o vėliau jis tobulinamas. To pasėkoje pasiekiamas lankstumas bendradarbiavimo tarp programuotojų, testuotojų ir administratorių pagrindu, o efektyvumas – dėl greitesnio ir kokybiško produkto pateikimo vartotojui.

Agile iteracijų ideologija užtikrina lankstumo ir efektyvumo suderinamumą. Kiekvienoje iteracijoje apibrėžiamas galutinis tikslas. Svarbu, kad jis būtų pasiekiamas ir jo sieks visa komanda. Iteracijos rezultatas bus pateikiamas vartotojui. Žinodami tikslą, komandos nariai patys pasiskirstys vaidmenis ir uždavinius, kas rodo lankstumą. Tikslas motyvuoja organizuoja ir disciplinuoja komandą, kas užtikrina kiekvienos iteracijos atlikimo efektyvumą. Fiksuotas iteracijos laikas moko programuotojus vertinti per kiek laiko jie gali atlikti vieną at kitą užduotį. Pagal tai lanksčiai gali būti paskirstytos užduotys tarp programuotojų, kad nebūtų vėlinimų, nes kiekvienas

programuotojas turi skirtingą patirtį, žinias, supratimą. Vėlinimų mažinimas lanksčiai paskirstant užduotis iteracijoje, kad tilpti į laiko rėmus, užtikrina programinės įrangos proceso efektyvumą.

4.2. Diskusija

Remiantis informantų interviu rezultatais, galima išskirti pagrindinius lankstumo principus, kurių taikymas užtikrintų Agile metodų taikymo efektyvumą.

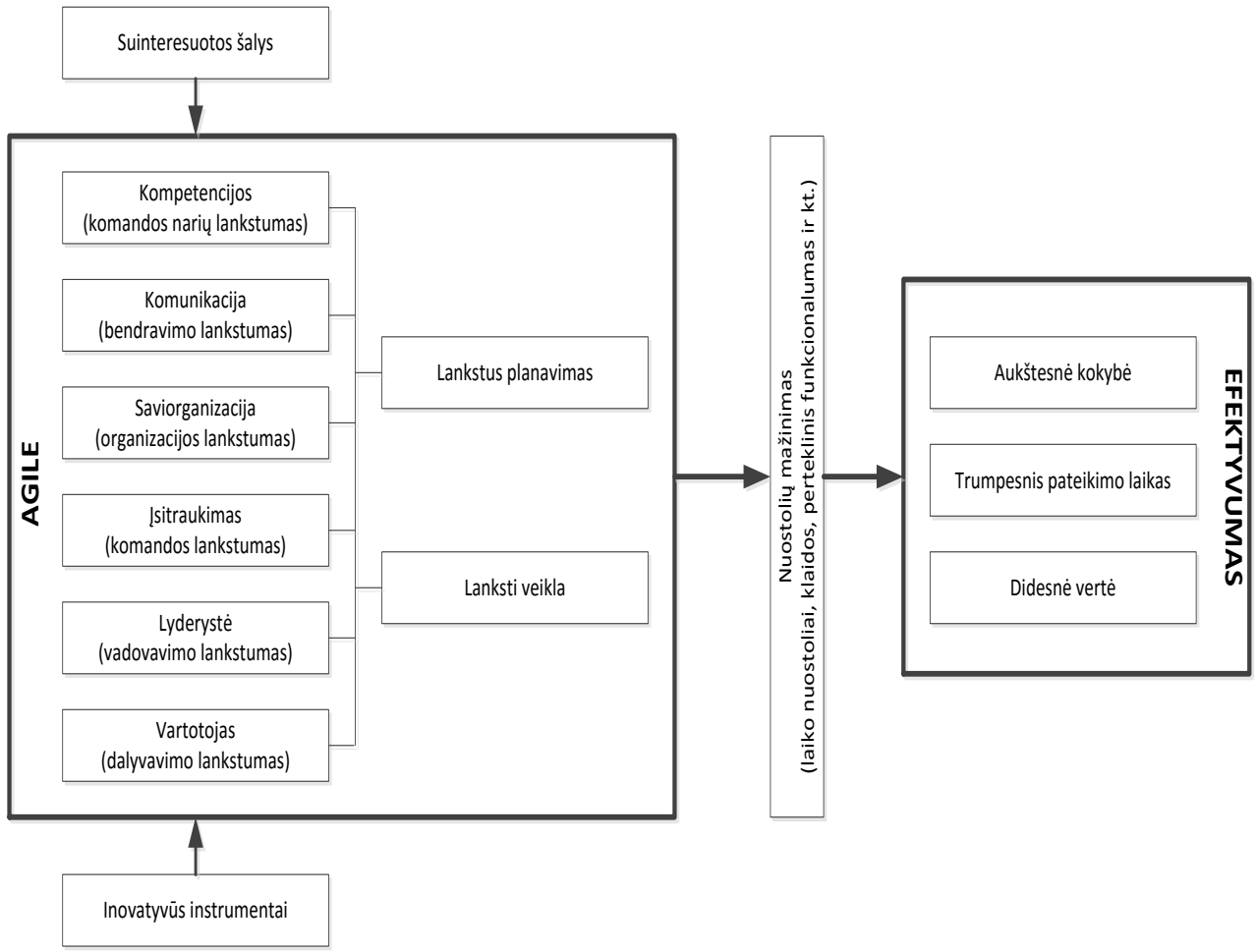
1. Lanksčios inovacijos. Be Agile metodų lankstumo tikslinga taikyti ir kitas inovatyvias technologijas. Viena iš jų gali būti taupios gamybos Lean technologija. Reikalinga mažinti besidubliuojančio kodo, bereikalingų ar besidubliuojančių funkcionalumų panaudojimą. Tai pasiekiama per komandos narių bendradarbiavimą ir bendravimą. Klaidų, dubliavimo skaičiaus sumažinimas sumažins ir laiko nuostolius, kas jau savo ruožtu padidina efektyvumą. Taip pat laiko nuostolius ir įvairias pertraukas projekto realizavime tikslinga taikyti DevOps technologiją, kuri pašalina nesutarimus ir konfliktus tarp programuotojų, testuotojų ir administratorių. Šios metodikos pagalba užtikrinamas ciklinis grįžtamasis ryšys iteracijų lygyje ir programuotojai gali operatyviau reaguoti į testuotojų ir administratorių pastabas, testuotojai – greičiau atlikti testavimą, o administratoriai – pašalinti nenaudojamų ar klaidingų funkcionalumų aptarnavimą. Pati Agile metodika, užtikrinanti lankstumą iteracijų pagalba, užtikrina ir efektyvumą, kuris pasiekiamas komunikacijos gerinimu ir laiko ribų iteracijai uždavimu.
2. Lankstus planavimas – leidžia efektyviai ir greitai reaguoti į aplinkos pokyčius. Tai užtikrinama vaidmenų perskirstymu komandoje, komandos saviorganizavimu, kuris leidžia greitai persiskirstyti vaidmenis ir užduotis pagal kliento poreikius. Kliento dalyvavimas taip pat užtikrina lankstų planavimą: vykdytojus greičiau pasiekia kliento poreikiai, o besimokanti komanda dažnai net gali numatyti būsimus kliento poreikius ir greitai į juos reaguoti. Reikia pabrėžti, kad planavimas turi vykti diskusijų forma: planas negali būti vieno asmens parengtas arba parengtas be konsensuso su vartotoju.
3. Lankstus vykdymas. Toks vykdymas leidžia komandai lengvai adaptuotis esamoje situacijose ir perskirstyti vaidmenis bei užduotis. Kiekvienas komandos narys žino galutinį tikslą ir jo siekia pagal savo kompetencijas, kurios žinomos ir kitiems komandos nariams. Tokiu būdu pasiekiama optimali ir efektyvi komandos organizacija, kuri užtikrina veikimą minimizuojant klaidas ir laiko praradimus, užtikrinant ne tik savo

veiklos kokybę, bet ir produkto kokybę. Visa tai yra efektyvumo sudėtinės dalys: mažinant klaidas ir lako nuostolius bei didinant kokybę, didėja veiklos efektyvumas.

4. Komandos lankstumas – jis užtikrinamas tuo, kad komandoje yra tarpusavio pakeičiamumas pagal kompetencijas. Komandoje taip pat yra keli svorio centrai, apie kuriuos gali koncentruotis atitinkamos iteracijos užduočių vykdymas. Pagrindinis svorio centras yra vartotojas, kurio reikalavimai ir dalyvavimas visuose programinio produkto kūrimo etapuose užtikrina efektyvią komunikaciją ir greitą reagavimą į vartotojo reikalavimus. Čia labai didelį vaidmenį vaidina efektyvi komunikacija. Tik dialogo pagalba gali būti priimti vartotojo keliami reikalavimai vykdymui. Priešingu atveju, vartotojas gali išgalvoti tokių reikalavimų, kurie stabdyti visą projekto realizavimo procesą.
5. Lanksti lyderystė – padeda komandos lyderiams priimti pagrįstus sprendimus, užtikrinančius efektyvų projekto realizavimo procesą, įgyti didesnį vartotojų pripažinimą, pagerinti komandos moralinę dvasią, o taip pat sumažinti nuostolius.

Apibendrintas Agile metodų lankstumo pagalba projekto realizavimo efektyvumo užtikrinimo modelis pateikiamas 1 pav.

Reikia pažymėti, kad atlikto tyrimo rezultatai patvirtina ir kai kurių mokslininkų teorines įžvalgas ir praktinius tyrimus. M. Omar ir kt. (2011) teigimu, teisingai panaudojant Agile metodus užtikrinamas programinės įrangos kūrimo efektyvumas ir investicijų atsiperkamumas. Geriausios Agile praktikos, didindamos darbo komandoje efektyvumą, užtikrindamos galimybę atlikti dažnus patikrinimus, saviorganizavimą ir atskaitomybę, o taip pat vidinį komandos narių lankstumą, užtikrina greitą ir efektyvų kokybiškos programinės įrangos pateikimą, kuris atitinka vartotojo keliamus reikalavimus. Tokiu būdu Agile metodas, kaip lankstus metodas, užtikrina tokius efektyvumo parametrus, kaip kokybę, pateikimo greitį ir kokybę, atitinkančią vartotojo reikalavimus.



4 pav. Lankstumo ir efektyvumo naudojant Agile metodus užtikrinimo modelis

Savo ruožtu C. Melo ir kt. (2013) apibrėžia pagrindines sąlygas, kuris išpildžius Agile metodų taikymas užtikrins programinės įrangos kūrimo efektyvumą:

1. Lyderio nustatymas vykdomajame lygyje. Bet kokia nauja iniciatyva, kuri liečia kritiškai svarbius komandos procesus, reikalauja ne tik palaikymo, bet ir proaktyvaus vadovo dalyvavimo. Šios iniciatyvos sėkmė priklauso nuo lyderio gebėjimų parodyti metodo taikymo privalumus konkrečiam uždaviniui spręsti.
2. Atsakomybės paskirstymas tarp komandos narių taip, kad kiekvienas būtų atsakingas už galutinį tikslą. Tai liečia ne tik galutinį tikslą (programinės įrangos sukūrimą), tačiau tikslą, iškeliamą kiekvienoje iteracijoje. Labai svarbu įvertinti ir kiekvieno komandos nario kompetencijas, kad reikalui esant bet kuriuo momentu vieną komandos narį būtų galima pakeisti kitu. Tokiu būdu šiame etape užtikrinamas komandos narių lankstumas. Taip eliminuojami laiko nuostoliai ieškant pakaitinio komandos nario.

3. Reikalinga išaiškinti komandos narių tarpusavio sąveikos svarbą. Informavimas yra tarpusavio sąveikos prielaida ir bendros komandos pagrindinis uždavinys. Svarbu, kad supratimas, jog būtina aktyvi tarpusavio sąveika, kokią naudą ji atneša kiekvienam komandos nariui ir kaip jam turi įtakos kasdieniai procesai. Tokiu būdu išaiškinama komunikacijos svarba ir kokią reikšmę komunikacija komandoje turi gerai suderintam komandos darbui.
4. Reikalinga sukurti nuoseklią metodiką su dideliu trumpos trukmės iteracijų skaičiumi. Informacinių technologijų aplinkoje, sukurtoje pagal tarpusavio sąveikos principą, tinkami žmonės naudoja tinkamas priemones tinkamu laiku, kas užtikrina galimybę daug kartų panaudoti pagrindinius išteklius, tokius kaip scenarijai, konfigūracija ir atskiros kodo dalys. Tokiu būdu užtikrinamas tinklinis ryšys tarp komandos narių, kas leidžia sumažinti laiko nuostolius dėl pakartotino kodo dalies rašymo ar klaidų taisymo. Tai ne tik leidžia sumažinti bereikalingas pastangas, bet ir pagerina apsikeitimą informacija.

Taip pat minėti autoriai pabrėžia, kad taikant Agile metodus efektyvumo padidinimą lemia ir papildomų instrumentų, pagerinančių komunikaciją tarp komandos narių, panaudojimo svarbą. Tai gali būti programinės priemonės, leidžiančios efektyviau komunikuoti, perduoti patirtį, dalintis informacija.

N. Moe (2010) atlikto tyrimo rezultatai parodė, kad taikant Agile metodiką, didžiausias efektyvumas kuriant programinius produktus pasiekiamas Scrum metodu, kurios komandą sudaro 7 (±2) asmenys. Komandos nariai turi efektyviai sąveikauti tarpusavyje. Komandoje nėra išankstinio pasiskirstymo vaidmenimis ir kiekvienas komandos narys turi gebėti pakeisti bet kurį kitą komandos narį. Atskirų komandos narių indėlis neturėtų būti vertinamas, kad nebūtų pažeidžiamas saviorganizacijos principas. Kiekvieno indelis vertinamas kaip vieningos komandos darbas. Pageidautina, kad komanda dirbtų vienoje patalpoje, o komandos nariai bet kuriuo metu galėtų matyti situaciją, kuri atvaizduojama vizualiai.

Apibendrinant galima pasakyti, kad atliktas tyrimas leido suformuoti Agile metodų lankstumo ir efektyvumo sąsajų modelį ir patvirtino, kad metodikos lankstumas, esant tinkamai parinktai komandai ir užtikrinus efektyvią komunikaciją tarp komandos narių, skatina komandos rezultatų efektyvumo didėjimą. Tačiau būtina įvertinti tai, kad efektyvumas pasiekiamas pritraukus vartotoją (kompetentingą jo atstovą) į komandos darbą. Efektyvumas padidėja, kai kartu su Agile metodais, naudojami ir kiti inovatyvūs instrumentai (pavyzdžiui, Lean metodika ar Kanban

metodą. Atliekant tolimesnius tyrimus, tikslinga būtų atlikti vartotojo dalyvavimo įtakos kuriamos programinės įrangos kokybei. Tai leistų išsiaiškinti, ar vartotojo keliami reikalavimai programinės įrangos kūrimo procese taikant Agile metodus nemažina šio metodo siekiamo efektyvumo.

IŠVADOS IR REKOMENDACIJOS

Išvados

1. Agile metodai taikomi neapibrėžtumo sąlygomis ir remiasi žmonių kūrybiškumu, o ne procesais. Jie charakterizuojami trumpais, pasikartojančiais prioritetinių atliekamų funkcijų ciklais, savianalizės ir apmastymo periodais, bendru sprendimų priėmimu, greito grįžtamo ryšio ir pokyčių įtraukimu, o taip pat nepertraukiamu programinės įrangos kodo pakeitimų integracija kūrimo stadijoje. Komanda dirba su nedidelėmis produkto dalimis, iš kurių kiekviena apima planavimą, kūrimą, integravimą, testavimą ir pateikimą. Programinio produkto kūrėjai dirba su klientais, kurie yra aktyvūs kūrimo proceso dalyviai, kartu vykdo prioritetines funkcijas, kas leidžia bendrai priimti sprendimus.
2. Agile metodai pateikia protingą programinės įrangos kūrimo būdą greitai besikeičiančiomis aplinkos sąlygomis. Tai pasiekiama Agile metodų principų pagalba. Jų teisingas panaudojimas atitinkamose aplinkybėse padeda sumažinti projekto rizikas, padidina projekto našumą ir galutinio produkto kokybę (pavyzdžiui, mažos komandos sumažina nekokybiškos komunikacijos riziką). Be to, kai pagrindiniai Agile metodų principai derinami su kitais Agile principais, gaunam sinergija, kuri užtikrina dar glaudesnį bendradarbiavimą projekte. Agile metodai projektų vadovams siūlo alternatyvią programinės įrangos kūrimo metodologiją ir valdymą, kas užtikrina tinkamą projektų palaikymą greitai besikeičiančių reikalavimų aplinkoje.
3. Atliktas tyrimas leido padaryti tokias išvadas:
   - Agile lankstumą ir efektyvumą projekto planavimo ir paruošimo etape užtikrina bendradarbiavimas su vartotoju. Lankstumas užtikrinamas tuo, kad bendradarbiaujant su vartotoju gaunami reikalavimai funkcionalumui (kokybei), vertei (projekto kainai) ir greičiui (programinės įrangos sukūrimo laikui). Visa tai užtikrina projekto realizavimo efektyvumą: projektas bus pateiktas su reikiamu funkcionalumu, tinkamais terminais ir už prieinamą kainą.
   - efektyvumas pasiekiamas ir tuo, kad kiekvienas projekto dalyvis įtraukiamas į projekto vykdymą visuose etapuose. Kiekvienas gali teikti savo pasiūlymus, į kuriuos bus atsižvelgta. Agile metodas numato visų programinės įrangos produkto kūrimo proceso dalyvių įtraukimą, paliekant jiems įprastas kompetencijas. Toks

būdas leidžia suprasti, kad jie visi dirba vienam ir tam pačiam galutiniam tikslui – sukurti kokybišką produktą savo klientui. Taikant Agile metodą, kiekvienas projekto dalyvis daro viską, todėl susiformuoja draugiškas kolektyvas, efektyviai veikiantis rinkoje. Paprastai taikant Agile metodą suformuojamos komandos, kurios dėl bendro tikslo ir bendros atsakomybės už viską, tampa emociniu kolektyvi, kuries dirba galutiniam tikslui – sukurti veikiantį kokybišką programinį produktą.

- lankstumas užtikrinamas ir per komandos saviorganizavimą. Tai leidžia atsisakyti bereikalingos vadybinės struktūros, nes egzistuoja tik vartotojo atstovas, kuris atstovauja verslo interesus. Be to, nebūtina tikrinti kiekvieną dalyvį: komanda pati pasiskirsto užduotis ir vidinę atsakomybę, o taip bus garantuotas darbo našumas ir produkto kokybė. Pati saviorganizacija yra komandos motyvavimas.

Rekomendacijos

1. Be Agile metodų lankstumo tikslinga taikyti ir kitas inovatyvias technologijas. Viena iš jų gali būti taupios gamybos Lean technologija. Reikalinga mažinti besidubliuojančio kodo, bereikalingų ar besidubliuojančių funkcionalumų panaudojimą. Tai pasiekiama per komandos narių bendradarbiavimą ir bendravimą. Klaidų, dubliavimo skaičiaus sumažinimas sumažins ir laiko nuostolius, kas jau savo ruožtu padidina efektyvumą.
2. Atsakomybę tarp komandos narių reikia paskirstyti taip, kad kiekvienas būtų atsakingas už galutinį tikslą. Tai liečia ne tik galutinį tikslą (programinės įrangos sukūrimą), tačiau tikslą, iškeliamą kiekvienoje iteracijoje. Labai svarbu įvertinti ir kiekvieno komandos nario kompetencijas, kad reikalui esant bet kuriuo momentu vieną komandos narį būtų galima pakeisti kitu. Tokiu būdu šiame etape užtikrinamas komandos narių lankstumas. Taip eliminuojami laiko nuostoliai ieškant pakaitinio komandos nario.

LITERATŪRA

1. Sfetsos, P.; Angelis, L.; Stamelos, I. (2006). Investigating the extreme programming system - An empirical study. *Empirical Software Engineering*, Vol. 11, Iss. 2, p. 269-301.
2. Vlaanderen, K., et al. (2011). The agile requirements refinery: Applying SCRUM principles to software product management. *Information and software technology* Vol. 53, Iss. 1, p. 58-70.
3. Livermore, J. (2008). Factors that significantly impact the implementation of an agile software development methodology. *Journal of Software*, Vol. 3, Iss. 4, p. 31-36.
4. Lucia, A.; Qusef, A. (2010). Requirements engineering in agile software development. *Journal of Emerging Technologies in Web Intelligence*, Vol. 2, Iss. 3, p. 212-220.Nerur, S, et al. (2010). Towards an understanding of the conceptual underpinnings of agile development methodologies. In: *Agile Software Development*. Berlin Heidelberg: Springer, p. 15-29.
5. Nerur, S.; Mahapatra, R.; Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM*, Vol. 48, Iss. 5, p. 72-78.
6. Kettunen, P.; Laanti, M. (2008). Combining agile software projects and large-scale organizational agility. *Software Process: Improvement and Practice*, Vol. 13, Iss. 2, p. 183-193.
7. Erdogmus, H. (2009). Architecture meets agility. *IEEE Software Magazine*, No. 5, p. 2-4.
8. Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. London: Addison-Wesley Professional
9. Dybå, T.; Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and software technology*, Vol. 50, Iss. 9, p. 833-859.
10. Baskerville, R, et al. (2006). High-speed software development practices: What works, what doesn't. *IT Professional*, Vol. 8, Iss. 4, p. 29-36.
11. Brooks, F. (1975). *The Mythical Man-Month: Essays on Software Engineering*. NY: Addison-Wesley
12. Naumann, J; Jenkins, A. (1982). Prototyping: the new paradigm for systems development. *MIS Quarterly*, No. 6, p. 29–44
13. Abbas, N.; Gravell,A.; Wills, G. (2008). Historical roots of Agile methods: where did "Agile thinking" come from?. In: *Agile Processes in Software Engineering and Extreme Programming*. Springer Berlin Heidelberg, p. 94-103.

14. Corbucci, H., et al. (2011). Genesis and Evolution of the Agile Movement in Brazil - Perspective from Academia and Industry. In *Proceedings of the 25th Brazilian Symposium on Software Engineering, SBES'11*, p. 98–107
15. Royce, W. (1987). Managing the development of large software systems. *Computer Society Press*, Vol. 2, Iss. 8, p. 1-9.
16. Larman, C.; Basili, V. (2003). Iterative and incremental development: A brief history. *IEEE Computer Society*, No. 36, p. 47–56, 2003
17. Naumann, J.; Jenkins, A. (1982). Prototyping: the new paradigm for systems development. *MIS Quarterly*, Vo. 6, p. 9–44.
18. Agile programinės įrangos kūrimo manifestas. [interaktyvus], [žiūrėta 2015 m. gruodžio 13 d.]. Prieiga per internetą: http://agilemanifesto.org/iso/lt/
19. Moe, N. (2011). *From Improving Processes To Improving Practice: Software Process Improvement in Transition from Plan-driven to Change-driven Development*. Oslo: Norwegian University of Science and Technology
20. Stober, T.; Hansmann, U. (2011). Overview of Agile Software Development. In: *Agile Software Development*. Springer Berlin Heidelberg, p. 35-59.
21. Popli, R.; Chauhan, N. (2013). Agile Software Development. *International Journal of Computer Science and Communication*, Vol. 4, Iss. 2, p. 153-156.
22. Cockburn, A.; Highsmith, J. (2010). *Agile software development: The people factor*. Computer, No. 34, p. 131–133.
23. Ford, D.; Voyer, J.; Wilkinson, J. (2010). Building learning organizations in engineering cultures: Case study. *Journal of Management in Engineering*, Vol. 16, Iss. 4, p. 72-83.
24. Salo, O.; Abrahamsson, P. (2007). An iterative improvement process for agile software development. *Software Process: Improvement and Practice*, Vol. 12, p. 1, p. 81–100.
25. Kettunen, P. (2009). *Agile software development in large-scale new product development*
26. *organization: team-level perspective*. PhD thesis, Helsinki University of Technology.
27. Lee, G.; Xia, W. (2010). Toward agile: An integrated analysis of quantitative and qualitative field data. *Management Information Systems Quarterly*, Vol. 34, Iss. 1, p. 87–114
28. Conboy, K.; Fitzgerald, B. (2004). Toward a conceptual framework of agile methods. In: *Extreme Programming and Agile Methods-XP/Agile Universe 2004*. Springer Berlin Heidelberg, p. 105-116.

29. Highsmith, J. (2004). *Agile Project Management - Creating Innovative Products*. Boston: Pearson Education, Boston.
30. Larman, C.; Basili, V. (2003). Iterative and incremental development: A brief history. *Computer*, No. 6, p. 47-56.
31. Erickson, J.; Lyytinen, K.; Siau, K. (2005). Agile modeling, agile software development, and extreme programming: the state of research. *Journal of database Management*, Vol. 16, Iss. 4, p. 88.
32. Henderson-Sellers, B.; Serour, M. (2005). Creating a dual-agility method: The value of method engineering. *Journal of Database Management*, Vol. 16, Iss. 4, p. 1-15.
33. Lyytinen, K.; Rose, G. (2006). Information system development agility as organizational learning. *European Journal of Information Systems*, Vol. 15, Iss. 2, p. 183-199.
34. Cockburn, A. (2006). *Agile software development: the cooperative game (agile software development series)*. London: Addison-Wesley Professional.
35. Qumer, A.; Henderson-Sellers, B. (2008). An evaluation of the degree of agility in six agile methods and its applicability for method engineering. *Information and software technology*, Vol. 50, Iss. 4, p. 280-295.
36. Greer, D.; Hamon, Y. (2011). Agile software development. *Software: Practice and Experience*, Vol. 41, Iss. 9, p. 943-944.
37. Ahalt, S., et al. (2014). Water Science Software Institute: Agile and open source scientific software development. *Computing in Science & Engineering*, Vol. 16, Iss. 3, p. 18-26.
38. Kurapati, N.; Manyam, V.; and Petersen, K. (2012). Agile software development practice adoption survey. In *Agile Processes in Software Engineering and Extreme Programming*, p.Berlin: Springer, 16–30.
39. Richard, P., et al. (2009). Measuring organizational performance: Towards methodological best practice. *Journal of management*, Vol. 35, Iss. 3, p. 718-804
40. Puškorius S. (2002). 3E koncepcijos plėtra. *Viešoji politika ir administravimas*, Nr. 3, p. 73-78
41. Mackevičius, J. (2007). *Įmonių veiklos analizė*. Vilnius: TEV
42. Dolan, S. (2006). *Managing by Values*. NY: Palgrave Macmillan
43. Drucker, P. (2008). *Efektyvaus vadovo veikla: žurnalas, padėsiantis padaryti tai, ką reikia*. Vilnius: Rgrupė
44. Mescon, M. (2007). *Excellence in Business*. London: Learning Solutions

45. Pyzdek, T.; Keller, P. (2009). *The Six Sigma Handbook*. London: McGraw-Hill Professional
46. Emiliani, M. (2006). Origins of lean management in America: The role of Connecticut businesses. *Journal of management History*, Vol. 12, Iss. 2, p. 167-184.
47. Poppendieck, M. (2011). Principles of lean thinking. *IT Management Select*, No. 18, p. 17-23.
48. Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. London: Addison-Wesley Professional
49. Droste, A. (2007). Lean thinking, banish waste and create wealth in your corporation, *Action Learning: Research and Practice*, Vol. 4, p. 105–106.
50. Poppendieck, M.; Poppendieck, T. (2007). *Implementing lean software development: From concept to cash*. London: Pearson Education.
51. Middleton, P.; Joyce, D. (2012). Lean Software Management: BBC Worldwide Case Study, *IEEE Transactions on Engineering Management*, Vol. 59. p. 20–32
52. Beck, L.; Andres, C. (2004). *Extreme Programming Explained: Embrace Change*. London: Addison-Wesley
53. Cao, L.; Ramesh, B. (2008). Agile requirements engineering practices: An empirical study. *Software*, Vol. 25, Iss. 1, p. 60-67.
54. Dybå, T.; Dingsoyr, T. (2009). What do we know about agile software development?. *Software*, Vol. 26, Iss. 5, p. 6-9.
55. Sutherland, J.; van Solingen, D. (2011). *The Power of Scrum*. London: CreateSpace Independent Publishing Platform
56. Sani, A., et al. (2013). A review on software development security engineering using dynamic system method (DSDM). *International Journal of Computer Applications*, Vol. 69, Iss. 25, p. 33-44.
57. Conboy, K., et al. (2011). People over process: key people challenges in agile developmen. *Software*, Vol. 28, Iss. 4, p. 48-57.
58. Mafakheri, F.; Nasiri, F.; Mousavi, M. (2008). Project agility assessment: an integrated decision analysis approach. *Production Planning and Control*, Vol. 19, Iss. 6, p. 567-576.
59. Agarwal, M.; Majumdar, E. (2012). Tracking Scrum projects Tools, Metrics and Myths About Agile. *International Journal of Emerging Technology and Advanced Engineering*, Vol. 2, Iss. 3, p. 97-104.

60. Petersen, K.; Wohlin, C. (2009). A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *Journal of systems and software*, Vol. 82, Iss. 9, p. 1479-1490.
61. Livermore, J. (2008). Factors that significantly impact the implementation of an agile software development methodology. *Journal of Software*, Vol. 3, Iss. 4, p. 31-36.
62. Pilone, D.; Miles, R. (2007). Head First Software Development. London: O'Reilly Media
63. Robillard, P. (2009). The role of knowledge in software development. Communications of the ACM, Vol. 42, Iss. 1, p. 87-92.
64. Unger, D.; Eppinger, S. (2011). Improving product development process design: a method for managing information flows, risks, and iterations. Journal of Engineering Design, Vol. 22, Iss. 10, p. 689-699.
65. Ruparelia, N. (2010). Software development lifecycle models. ACM SIGSOFT Software Engineering Notes, Vol. 35, Iss. 3, p. 8-13.
66. Wautelet, Y.; Kolp, M.; Poelmans, S. (2013). Requirements-driven iterative project planning. In: Software and Data Technologies. Berlin: Springer, p.. 121-135.
67. Ragunath, P., et al. (2010). Evolving a new model (SDLC Model-2010) for software development life cycle (SDLC). International Journal of Computer Science and Network Security, Vol. 10, Iss. 1, p. 112-119.
68. France, R.; Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. Environmental Modelling & Software, Vol. 21, Iss. 5, p. 602-614.
69. Beck, K.; Andres, C. (2004). Extreme Programming Explained: Embrace Change. London: Addison-Wesley
70. Bosch-Sijtsema, K. et al.(2009) Knowledge work productivity in distributed teams. Journal of Knowledge Management, Vol. 13, Iss. 6, p. 533–546
71. Feyh, M.; Petersen, K. (2013). Lean Software Development Measures and Indicators-A Systematic Mapping Study, In: Lean Enterprise Software and Systems, Springer, p. 32–47.
72. Clarke, P.; O'Connor, R. (2012). The situational factors that affect the software development process: Towards a comprehensive reference framework. Information and Software Technology, Vol. 54, Iss. 5, p. 433-447.
73. Petersen, K. (2011). Measuring and predicting software productivity: A systematic map and review. Information and Software Technology, Vol. 53, Iss. 4, p. 317-343.

74. Poppendieck, M.; Poppendieck, T. (2006). Implementing Lean Software Development: From Concept to Cash. London: Addison-Wesley Professional
75. Omar, M.; Syed-Abdullah, S.; Yasin, A. (2011). The impact of agile approach on software engineering teams. American Journal of Economics and Business Administration, Vol. 3, Iss. 1, p. 12-27.
76. Melo, C.; Cruzes, D.; Kon, F.; Conradi, R. (2013). Interpretative case studies on agile team productivity and management. Information and Software Technology, Vol. 55, Iss. 2, p. 412-427.
77. Moe, N.; Dingsøyr, T.; Dybå, T. (2010). A teamwork model for understanding an agile team: A case study of a Scrum project. Information and Software Technology, Vol. 52, Iss. 5, p. 480-491.

PRIEDAI

1 PRIEDAS. Interviu gidas

Modulis	Tema	Klausimai
Agile metodas įmonėje	Agile diegimo motyvai	1. Jei žinote, kada Agile metodai pradėti taikyti jūsų įmonėje? Gal žinote, kodėl įmonė ėmė taikyti Agile?
	Taikoma Agile atmaina	2. Kokį Agile metodą Jūs naudoja jūsų įmonė: Agile-Lean, Agile XP, Agile SCRUM, Dinaminis sistemų kūrimo metodą ar kokį kitą? Jei žinote, kodėl buvo pasirinktas šis?
	Taikymo sritis	3. Jei žinote, kokiai Jūsų vykdomų projektų daliai naudojate Agile metodus? Kokio tipo programinės įrangos kūrimo projektams naudojate Agile metodus? (pavyzdžiui, Interneto Duomenų apdorojimo, Vartotojo sąsajos, Atskira programa, Verslo valdymo sistema, Realaus laiko/Valdymo, Sistemos (OS ir pan.)
Agile projekto įgyvendinimas	Projekto vyksmas	4. Gal galite kiek įmanoma detaliau papasakoti kaip vyksta projekto įgyvendinimas taikant Agile, remiantis konkrečiu projektu?
	Su Agile metodu siejamos darbo priemonės	5. Kokias programines priemones naudojote programinių priemonių kūrimo projektuose, taikant Agile metodą?
	Su Agile siejamos organizacinės priemonės	6. Kokios organizacinės priemonės buvo taikomos naudojant Agile metodą?
	Agile metodo taikymo problemos	7. Su kokiomis problemomis susidūrėte naudodami Agile programinės kūrimo metodą?
Metodų/praktikų taikymo privalumai ir trūkumai	Agile metodo taikymo privalumai	8. Kokias teigiamas pasekmes teikia dažniausiai Jūsų įmonėje naudojamo Agile metodo taikymas?

	Agile metodo taikymo trūkumai	9. Kokias neigiamas pasekmes teikia dažniausiai Jūsų įmonėje naudojamo Agile metodo taikymas, jei tokių yra?
Vartotojo vaidmuo	Vartotojo dalyvavimo programinės įrangos kūrime laipsnis	10. Koks kliento dalyvavimo laipsnis Agile programinės įrangos kūrime?
	Vartotojo dalyvavimas programinės įrangos kūrimo procese	11. Apibūdinkite kliento dalyvavimą programinės įrangos kūrimo procese
Agile metodo taikymo efektyvumas	Programinės įrangos kūrimo taikant Agile metodą efektyvumo kriterijai	12. Kokius efektyvumo kriterijus Jūs naudojate vertindami programinės įrangos kūrimo procesą?
	Programinės įrangos kūrimo taikant Agile metodą efektyvumo užtikrinimas	13. Kaip jūs užtikrinate efektyvų programinės įrangos kūrimo procesą?
	Vartotojo vaidmuo užtikrinant efektyvumą	14. Koks vartotojo vaidmuo vertinant programinės įrangos kūrimo proceso efektyvumą?
Lankstumo užtikrinimas taikant Agile metodus	Lankstumo užtikrinimas	15. Kaip užtikrinamas programinės įrangos kūrimo proceso lankstumas taikant Agile metodus?
	Vartotojo vaidmuo lankstumo užtikrinime	16. Koks vartotojo vaidmuo užtikrinant programinės įrangos kūrimo proceso lankstumą?
Agile metodo taikymo lankstumo ir efektyvumo santykis	Lankstumo ir efektyvumo santykis	17. Kaip Jūs vertinate Agile programinės įrangos kūrimo metodo lankstumo ir efektyvumo santykį?