VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Programų kūrimo proceso judrumo gerinimo priemonės

Software process improvement measures of agility

Bakalauro darbas

Atliko: 4 kurso 4 grupės studentė
Gabrielė Gelumbauskaitė
(parašas)

Darbo vadovas: Lekt. Andrius Adamonis
(parašas)

Darbo recenzentas: Asist. Stasys Peldžius
(parašas)

Vilnius – 2016

Santrauka

Pastaruoju metu įmonės daug dėmesio skiria programų kūrimo proceso gerinimui. Tam pasitelkiamos įvairios metodologijos, aprašančios gerąsias praktikas, kurias komandos turėtų taikyti norint pasiekti aukštesnį judrumo lygį. Gerindamos projekto kūrimo procesą įmonės didina proceso judrumo lygį. 2014 metais buvo sukurta pirmoji tolydinio judrumo vertinimo modelio struktūra, kurioje aprašomos judrumo ir aspektų dimensijos. 2015 metais ši struktūra buvo papildyta aspektų dimensijai priskiriant gerąsias praktikas.

Bakalaurinio darbo metu buvo atlikta apklausa, tiriant aspektų dimensijai priskirtų judriųjų praktikų naudojimą realiose įmonėse. Apklausos tikslas - ištirti, ar įmonės naudoja aspektų dimensijai priskirtas judriųjų praktikų grupes bei ar tos pačios praktikos vienodai tinkamos naudoti ir komandoms kuriančioms naujas sistemas, ir prižiūrinčioms senas. Jei komandos praktikas naudoja ne pilnais rinkiniais buvo bandoma išsiaiškinti ar tikrai ta praktika turi būti įtraukta į judrumo vertinimo modelį.

Raktiniai žodžiai: proceso judrumo vertinimo modelis, proceso judrumo vertinimas, kuriančios sistemas komandos, prižiūrinčios sistemas komandos, judriosios praktikos, aspektų dimensija.

Summary

Companies has recently increased focus on software development process improvement. They are utilizing various agile methodologies to reach a higher agility level. By increasing agility level, teams improve software development process in the organization. In 2014, the first continuous process agility assessment model structure was implemented which described agility and aspect dimensions. In 2015, that structure was supplemented by assigning agile practices to aspect dimensions.

As a part of this writing, a survey has been conducted to research usage of agile practices assigned to aspect dimensions in companies. The purpose of the survey was to examine if companies used agile practice groups assigned to aspect dimensions and if those practices were also used in teams which developed new systems and supported old ones. If teams used only part of a full practice set, an attempt was made to find out if those skipped practices should be included into a process agility model.

Keywords: process agility model, process agility assessment, developing team, maintaining teams, agile practices.

Turinys

Įvadas

Iki 2000m. buvo svarstoma ką galima pakeisti įmonės darbe, kad sumažėtų nepasisekusių projektų skaičius, bei sukurta nemažai metodologijų (Scrum 1986m., Extreme programming 1996m. ir kt.), tačiau jos nebuvo plačiai taikomos. Situacija pasikeitė 2001m., kai 17 skirtingų sričių specialistų pasirašė judrumo manifestą (*ang. agile manifesto*), kuriame išdėstyti kriterijai judriai įmonei[AM01]. Išleidus manifestą sukurta nemažai naujų bei patobulintos senos metodologijos atsižvelgiant į manifeste apibrėžtus programinės įrangos principus.

Visos metodologijos turi tam tikras praktikas, kurias pritaikius programinės įrangos kūrimas turėtų būti judresnis, tačiau dažniausiai įmonės paredaguoja praktikas pagal savus poreikius, laikosi ne visų praktikų, todėl jų judrumą nustatyti pasidaro labai sunku. Kol kas nėra sukurto oficialaus, visuotinai pripažinto modelio programų kūrimo proceso judrumo lygiui nustatyti, tačiau pastaruoju metu bandoma tokį modelį sukurti.

2014m. mokslininkai Özden Özcan Top ir Onur Demirors[OD14] pradėjo tirti judriojo programavimo vertinimo lygius ir apie tai parašė straipsnį „Programinės įrangos judrumo vertinimas: mokslinis tyrimas“ (angl. *Assessing Software Agility: An Exploratory Case Study*). Mokslininkų aprašytame tolydiniame proceso judrumo vertinimo modelyje pateikta modelio struktūra, bet ne pilnas modelis. Modelį sudarė dvi dimensijos - judrumo ir aspektų. Aspektų dimensiją sudaro penki lygiai, tačiau mokslininkų darbe nebuvo aprašyta, kokios judriosios praktikos įeina į šiuos lygius.

2015m. Vilniaus universiteto magristrantas Nerijus Mazėtis atliko tiriamąjį darbą „Proceso judrumo vertinimo modelis“[Ner15]. Darbe jis atrinko įvairių metodologijų judriąsias praktikas ir, pagal prasmę, sugrupavo jas į aspektų lygius. Aspektui priskirtos judriosios praktikos turi pagerinti įmonės veiklą aspekto atžvilgiu.

Šio darbo tikslas - patvirtinti arba paneigti hipotezę, kad proceso judrumo vertinimo modelyje aprašytos judriosios praktikos tinkamos taikyti tiek programų sistemų kūrimo, tiek priežiūros procesuose ir kad jų sugrupavimas į aspektus atitinka realią įmonių praktiką.

Darbo uždaviniai:

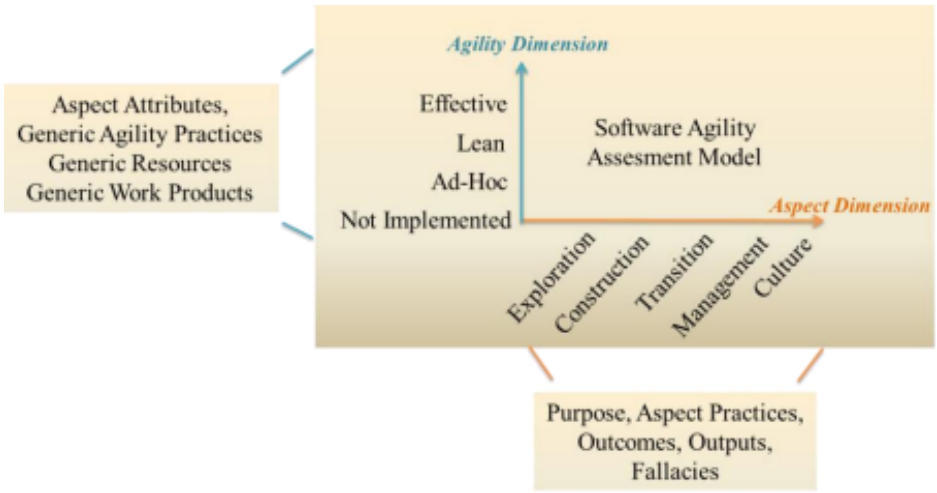
- Patikrinti, ar įmonės, savo veikloje, taiko proceso judrumo vertinimo modelio aspektų dimensijai priskirtas judriųjų praktikų grupes.
- Patikrinti, ar judrumo vertinimo modelyje aprašytos judriosios praktikos gali būti taikomos ir programų kūrimo, ir priežiūros procesuose.

Darbo gale pateikiamas pakoreguotas proceso judrumo vertinimo modelis, kuriame išskirti programų sistemų kūrimo bei priežiūros procesai.

1. Proceso judrumo vertinimo modelis

1.1. Proceso judrumo vertinimo modelio struktūra

Mokslininkai Özden Özcan Top ir Onur Demirors straipsnyje „Programinės įrangos judrumo vertinimas: mokslinis tyrimas“ apibrėžė programinės įrangos kūrimo judrumo vertinimo modelio (*angl. the Software Agility Assessment Model*) struktūrą pagal ISO/IEC 15504 modelio konstrukciją, tačiau vietoje gebėjimo ir procesų dimensijos autoriai aprašo judrumo ir aspektų dimensijas (žr. 1 pav.)[OD14,ISO04] . Šis modelis yra pirmasis tolydinis judrumo vertinimo modelis.



1 pav. programinės įrangos judrumo vertinimo modelis

Aspektų dimensija - pagal prasmę sugrupuotos judriųjų metodikų praktikos. Aspektų dimensiją sudaryta iš:

- tyrinėjimo (angl. *Exploration*) - reikalavimų tyrimas ir valdymas;
- konstravimo (angl. *Construction*) - projektavimas, programavimas ir modulių testavimas;
- perėjimo (angl. *Transition*) - integravimas, diegimas, testavimas;
- valdymo (angl. *Management*) - darbų planavimas, vertinimas, testavimas;
- kultūros (angl. *Culture*) - darbuotojams sąlygų sudarymas

Kokias konkrečiai praktikas įtraukė į kiekvieną aspektą, autoriai nenurodė.

Judrumo dimensija - judrumo manifeste aprašytų judriųjų metodikų principų atitikimo vertinimas[AM01] (žr. Lentelė nr. 1).

1 lentelė. Judrumo lygiai

Judrumo dimencijos	**Agile manifesto principai**
Neįgyvedintos (angl. *Not implemented*)	-
Nepasiteisinusios (angl. *Ad-Hoc*)	-
Liesos (angl. *Lean*)	1, 2, 3, 4, 6, 7, 10
Efektyvios (angl. *Effective*)	5, 8, 9, 11, 12

1.2. Proceso judrumo vertinimo modelis

Nerijus Mazėtis tiriamajame darbe „Proceso judrumo vertinimo modelis“[Ner15] detalizavo Mokslininkų Özden Özcan Top ir Onur Demirors aprašytą modelį. Atlikęs tyrimą pasirinko tris metodologijas (XP, Scrum bei DSDM Atern), sugrupavo jose aprašomas judriąsias praktikas ir priskyrė jas atitinkamoms aspektų dimensijoms (2 lentelė). Judriųjų praktikų, kurios taikomos keliose metodologijos, pavadinimai pakeisti siekiant nepririšti jų prie metodologijos.

2 lentelė. Kokybinis judriųjų praktikų ir aspektų gretinimas

Aspektas	**Judriosios praktikos**
Tyrinėjimo	Vartotojų pasakojimai
	Evoliuciniai reikalavimai
	Modeliavimas ir prototipavimas
Konstravimo	Kodo pertvarkymas
	Modulių testai
	Porinis programavimas
	Bendra kodo nuosavybė
	Kodo standartai
	Testavimu grįstas kūrimas
Perėjimo	Dažnos laidos
	Nuolatinė integracija
	Judrusis testavimas
Valdymo	Darbų sąrašas
	Planavimo žaidimas
	Vizuali ir prieinama projekto informacija
	Judrusis projektų darbų apimties vertinimas

	Komandos susirinkimai
Kultūros	Užsakovo įtraukimas į kūrimo procesą
	Jokių viršvalandžių
	Fizinė aplinka
	Saviorganizuojančios komandos

2. Tyrimo metodika

Pagrindinis mokslininkų Özden Özcan Top ir Onur Demirors sukurto judrumo vertinimo modelio išskirtinumas - tai pirmasis tolydinis judrumo vertinimo modelis. Tolydinis modelis leidžia organizacijai pasirinkti proceso sritis ir siekti užsibrėžto jų gebėjimo lygio [SEI07] (pavyzdžiui, pagal mokslininkų Özden Özcan Top ir Onur Demirors aprašytą judrumo modelį, jeigu įmonė užsibrėžia pagerinti savo tyrinėjimo aspekto lygį, tai ji turi pritaikyti visas tyrinėjimo aspektui priklausančias judriąsias praktikas).

Norint išsiaiškinti, ar proceso judrumo modelyje aspektų dimensijai priskirtas judriąsias praktikas įmonės taiko savo veikloje, buvo atliktas tyrimas. Tyrimo metu buvo sudarytas klausimynas, kuriame užduodami klausimai, kokias judriąsias praktikas naudoja komanda ir ar ji kuria, ar prižiūri projektą. Klausimai buvo atrinkti pagal Nerijaus Mazėčio proceso judrumo vertinimo modeliui priskirtas judriąsias praktikas. Kiekvienai judriąjai praktikai, priskirtai proceso vertinimo modelio aspektui, buvo sukurta po klausimą arba kelis klausimus siekiant kuo aiškiau suprasti ar komanda naudoja šią judriąją praktiką. Anketos tikslas - patikrinti, ar komandos, savo veikloje taikančios praktikas, jas išpildo pilnais, aspektų dimencijai priskirtais, rinkiniais. Įmonės, norinčios turėti judrų procesą dažniausiai taiko įvairias judriąsias praktikas padedančias tai pasiekti, todėl, jei įmonės taiko daugumą aspektui priklausančių judriųjų praktikų, tačiau dauguma įmonių išskiria ir netaiko konkrečios aspektui priklausančios praktikos/praktikų, tai dėl tam tikrų priežasčių šią/šias praktikas įmonėms taikyti neapsimoka. Darbe analizuojama ar tokias praktikas reikia įdėti į judrumo vertinimo modelį ir jos netaikomos dėl to, kad įmonėms prireiktų laiko pradėti taikyti šią praktiką, nors praktika atneštų naudos, ar šios praktikos/praktikų nereikėtų įtraukti į proceso judrumo vertinimo modelį, nes nauda taikant šią praktiką neatsvertų laiko ir kainos reikalingos praktikos taikymui.

Darbe siekiama išsiaiškinti, ar proceso judrumo vertinimo modelis tinka tiek komandoms kuriančioms, tiek prižiūrinčioms sistemą, todėl išskiriamos ir tos judriosios praktikos, kurias taiko ženkliai skirtingas skaičius kuriančių sistemą komandų ir prižiūrinčių sistemą. Šios judriosios praktikos analizuojamos taip pat, kaip ir tos, kurios apskritai nėra taikomos (taiko mažuma komandų).

Klausimyną užpildė 53 asmenys dirbantys skirtingose komandose. 36 anketą atsakiusios komandos prižiūri egzistuojančią sistemą, 28 - kuria naujas sistemas (kai kurios komandos ir kuria ir prižiūri sistemas).

Pabaigus darbą tikimasi gauti naują proceso vertinimo judrumo modelį, kuriame išskirtos proceso kūrimui bei priežiūrai tinkamos judriosios praktikos bei įtrauktos tik tos praktikos, kurias įmonės naudoja arba turėtų naudoti savo procesuose.

3. Tyrinėjimo aspektas

3 lentelė. Tyrinėjimo aspekto nagrinėjimas

Judrioji praktika	**Aprašymas**	**Užduotas klausimas**
Vartotojų pasakojimai (angl. User stories)	Judrioji praktika, kuomet reikalavimai produktui yra formuluojami vienu arba dviem sakiniais užsakovui ir vykdytojui suprantama kalba. Vartotojo pasakojimai paskui transformuojami į konkrečias programavimo užduotis bei naudojami kuriant testus. Vartotojo pasakojimai pakeičia tradicinėse metodologijose naudojamas dideles ir formalias reikalavimų specifikacijas.	Ar turite reikalavimų specifikacijos dokumentą? *Pagal aprašymą, komandos, kurios nenaudoja reikalavimų specifikacijos dokumento, praktikuoja vartotojų pasakojimų judriąją praktiką.*
Evoliuciniai reikalavimai	Reikalavimai turi būti renkami dalimis, o ne vienu katu. Pavyzdžiui, projektuojant produktą turi būti apibrėžiami tik pagrindiniai reikalavimai. Vėlesnėse iteracijose reikalavimai detalizuojami, atsiranda nauji. Tai leidžia sprendimus priimti atsižvelgiant į kliento atsiliepus apie jau sukurtą funkcionalumą.	Kuriuo metu surenkami reikalavimai sistemai?
Modeliavimas ir prototipavimas	Judriosiose metodikose vizualizacija siekiama gerinti bendravimą. Modeliai padeda analizuoti bei dokumentuoti reikalavimus, spręsti problemas. Detalizuoti reikalavimai užsakovui demonstruojami prototipais. Prototipų demonstravimas išplečia naudotojo supratimą apie sistemos galimybes ir yra tinkamas sužinoti, kaip naudotojai vertina sistemos korektiškumą, tinkamumą naudoti, ir nustatyti užsakovo poreikiams.	Ar kuriate prototipus?

Norint atrinkti tik tas komandas, kurios linkusios taikyti tyrinėjimo aspekto judriąsias praktikas buvo atrinkta praktika, kuri bus laikoma esmine tyrinėjimo aspekto praktika. Atskirai pateikiami procentai komandų kuriančių naujas sistemas ir prižiūrinčių senas sistemas.

Duomenys komandų kuriančių sistemą:

Kuriuo metu surenkami reikalavimai sistemai? Reikalavimai atsiranda arba kinta projekto eigoje: 96.4%

Duomenys komandų prižiūrinčių sistemą:

Kuriuo metu surenkami reikalavimai sistemai? Reikalavimai atsiranda arba kinta projekto eigoje: 94.4%

Atrinkus tik tas komandas, kurios taiko visas šias judriąsias praktikas buvo iš naujo perskaičiuoti visų konstravimo aspektui priklausančių judriųjų praktikų taikymo duomenys (4 lentelė).

4 lentelė. Tyrinėjimo aspekto judriųjų praktikų taikymas

Klausimas:	**Komandos kuriančios sistemas**	**Komados prižiūrinčios sistemas**
Ar turite reikalavimų specifikacijos dokumentą?	Taip: 17% Ne: 83%	Taip: 23% Ne: 77%
Ar kuriate prototipus?	Taip: 54% Ne: 46%	Taip: 16% Ne: 84%

Pagal judriosios praktikos „Vartotojų pasakojimai“ apibrėžimą įmonės naudojančios šią praktiką neturi reikalavimų specifikacijos dokumento. Tokiu atveju ~80% ir kuriančių, ir prižiūrinčių sistemas komandų naudoja šią praktiką. Tai yra pakankamas procentas, kad būtų galima laikyti, jog ši praktika yra naudinga komandoms ir reikalinga tyrinėjimo aspektui.

Praktiką „Modeliavimas ir prototipavimas“ taiko dauguma kuriančių naują sistemą komandų, tačiau mažuma prižiūrinčių sistemas komandų, todėl ši praktika plačiau bus aptarta poskyriuje „Modeliavimas ir prototipavimas“.

3.1. Modeliavimas ir prototipavimas

Pliusai[Ne04,MSP13]:

- Prototipų kūrimas panaikina nesusikalbėjimą su užsakovu ankstyvoje stadijoje, todėl galutinės versijos būna pigesnės ir greičiau padaromos.
- Užsakovas yra labiau įtrauktas į kūrimo procesą.
- Sunkiai suprantamas arba klaidinantis funkcionalumas greitai aptinkamas ir ištaisomas.
- Greitai aptinkami neaprašytas, trūkstamas reikalavimų dokumente, funkcionalumas.

- Sistemos kuriamos naudojant prototipavimą turi geresnę architektūrą, nes sukūrus prototipus yra aišku, kaip sistema turi veikti ir užsakovo poreikiai dažniausiai keičiasi labai minimaliai.
- Klaidas, pastebėtas kuriant/sukūrus prototipą, pataisyti žymiai pigiau ir lengviau nei klaidas rastas sistemos kūrimo proceso pabaigoje.
- Kai yra sukurtas prototipas, lengviau planuoti užduočių laiką.

  Minusai[Ne04,]:

- Didelės pradinės laiko išlaidos. Nors ilgalaikėje perspektyvoje prototipavimas atsiperka, tačiau darant mažus projektus, kuriems labai svarbu yra greitis, prototipavimas gali būti netinkamas.
- Prototipavimo esmė - jis turi būti padarytas greitai, tik su esminiu funkcionalumu. Dažnai programuotojai pamiršta šį faktą ir prototipus daro gana detalius, su idėja, kad vėliau, kūrimo fazėje jie tęs šį darbą. Taip gaištamas laikas ir prototipavimas praranda esmę.
- Dažnai programuotojai, sukūrę prototipus, nori toliau kurti ir galutinę sistemą ant prototipų, nors jų achitektūra tam nėra tinkama.
- Dažnai užsakovai, pamatę prototipus nori, kad darbas būtų tęsiamas toliau, nors prototipo architektūra netinkama galutiniam projektui.

3.2. Išvados

Prototipavimas yra gera praktika komandoms kuriančioms naują sistemą, nes ją naudojant lengviau ir tiksliau išsiaiškinami reikalavimai, pakeitimai daromi ankstyvoje kūrimo fazėje, todėl yra lengviau padaromi bei pigesni.

Komandoms prižiūrinčioms sistemas ši praktika visiškai nėra taikoma, todėl proceso judrumo vertinimo modelyje reikėtų pridėti sąlygą, kad ši judrioji praktika skirta tik komandoms kuriančioms naujas sistemas.

4. Konstravimo aspektas

5 lentelė. Konstravimo aspekto nagrinėjimas

Judrioji praktika	**Aprašymas[NER15]**	**Užduotas klausimas**
Kodo pertvarkymas (angl. Refactoring)	Programos kodo keitimas nekeičiant jos funkcionalumo. Dažniausiai kodas pertvarkomas siekiant pagerinti jo skaitomumą, supaprastinti struktūrą. Kuriant programinę įrangą iteracijomis, kodo pertvarkymas yra viena iš svarbiausių praktikų.	Ar disciplinuotai/rutiniškai taikote kodo pertvarkymo praktiką (angl. Code refactoring)?
Modulių testai (angl. Unit Test)	Atskirų programinės įrangos komponentų (klasių, ar funkcijų), izoliuotų nuo likusios sistemos, testai. Paprastai tokie testai kuriami prieš pradedant programuoti, automatizuojami, o radus naują defektą, kuriami papildomi testai. Prieš integruojant kodą, rekomenduojama jį patikrinti modulių testais.	Ar taikomas automatinis testavimas? Kiek procentų sistemos padengia automatiniai testai? Automatinių testų praktikos: Modulių testavimas Sąsajų integracijos testavimas Našumo testavimas
Porinis programavimas (angl. Paired programming)	Judrioji praktika, kuomet kūrėjai dirba poromis, tikrinant vienas kito darbą ir teikiant paramą visada gerai atlikti užduotį.	Ar komandoje taikoma porinio programavimo praktika?
Bendra kodo nuosavybė	Bendra kodo nuosavybė reiškia, kad kiekvienas projekte dalyvaujantis asmuo gali keisti bet kurią kodo dalį.	Ar praktikuojamos kodo peržiūros (angl. Code review)? Kodo peržiūrų (angl. Code review) tikslai: Kodo korektiškumas funkcionalumo prasme

		Tvarkingas ir suprantamas kodas Vieningo kodo standarto laikymasis Apsikeitimas žiniomis Perdavimas priežiūrai/palaikymui
Kodo standartai	Kodas turi būti parašytas vienodai, kad būtų lengvai skaitomas ir pertvarkomas kitų.	Ar laikomasi kodo standarto?
Testavimu grįstas kūrimas (angl. Test driven development)	Programinės įrangos kūrimo praktika, kuomet pirma parašomi modulių testai ir tuomet rašomas kodas. Vėliau šie testai gali būti naudojami testavimo automatizavimui.	Ar taikomas automatinis testavimas? Ar praktikuojamas testavimu grįstas kūrimas (angl. Test driven development)?

Norint atrinkti tik tas komandas, kurios linkusios taikyti konstravimo aspekto judriųjų praktikų buvo atrinktos praktikos, kurių laikosi dauguma komandų. Atskirai pateikiami procentai komandų kuriančių naujas sistemas ir prižiūrinčių senas sistemas.

Duomenys komandų kuriančių sistemą:

- Ar laikomasi kodo standarto: 92.8%
- Ar praktikuojamos kodo peržiūros (angl. Code review)?: 78.6%
- Ar taikomas automatinis testavimas?: 60.7%

Komandos prižiūrinčios sistemą:

- Ar laikomasi kodo standarto: 86.1%
- Ar praktikuojamos kodo peržiūros (angl. Code review)?: 77.4%
- Ar taikomas automatinis testavimas?: 64.2%

Atrinkus tik tas komandas, kurios taiko visas šias judriąsias praktikas buvo iš naujo perskaičiuoti visų konstravimo aspektui priklausančių judriųjų praktikų taikymo duomenys (žr. 5 lentelė).

6 lentelė. Konstravimo aspekto judriųjų praktikų taikymas

Klausimas:	**Komandos kuriančios sistemas**	**Komados prižiūrinčios sistemas**
Kiek procentų sistemos padengia automatiniai testai?	85%+: 25% 51-85%: 37.5% 16-50%: 31.3% 0-15%: 6.3%	85%+: 26.3% 51-85%: 31.6% 16-50%: 36.8% 0-15%: 5.3%
Automatinių testų praktikos:	Modulių testavimas: 75% Sąsajų integracijos testavimas: 75% Našumo testavimas: 25%	Modulių testavimas: 45% Sąsajų integracijos testavimas: 67.5% Našumo testavimas: 25%
Ar komandoje taikoma porinio programavimo praktika?	Taip: 31.2% Ne: 68.8%	Taip: 36.8% Ne: 63.2%
Kodo peržiūrų (angl. Code review) tikslai:	Kodo korektiškumas funkcionalumo prasme:87.5% Tvarkingas ir suprantamas kodas: 100% Vieningo kodo standarto laikymasis: 87.5% Apsikeitimas žiniomis: 81.5% Perdavimas priežiūrai/palaikymui: 53.25%	Kodo korektiškumas funkcionalumo prasme:85% Tvarkingas ir suprantamas kodas: 92.5% Vieningo kodo standarto laikymasis: 80% Apsikeitimas žiniomis: 70% Perdavimas priežiūrai/palaikymui: 40%
Ar praktikuojamas testavimu grįstas kūrimas (angl. Test driven development)?	Taip: 66.3% Ne: 33.7%	Taip: 26.3% Ne: 73.7%

Virš 50% kuriančių ir prižiūrinčių sistemą komadų padengia automatiniais testais daugiau nei 50% sistemos. Modulių testavimą taiko 75% kuriančių sistemą komandų, tačiau tik 45% prižiūrinčių sistemą komandų, todėl ši praktika bus aprašyta atskirai poskyriuje „Modulių testavimas“.

Praktika „Bendra kodo nuosavybė“ yra išpildoma, nes visos atrinktos komandos laikosi kodo peržiūrų praktikos ir virš 80% komandų, tai daro dėl to, kad kodas būdų tvarkingas ir suprantamas arba atitiktų kodo standartą.

Porinio programavimo praktikos netaiko daugiau nei 60% kuriančių bei prižiūrinčių komandų, todėl ji bus aptarta poskyriuje „Porinis programavimas“.

Testavimu grįsto kūrimo praktikos laikosi labai skirtingas procentas kuriančių nuo prižiūrinčių sistamą komandų, todėl ši praktika aptariama skyriuje „Testavimu grįstas kūrimas.

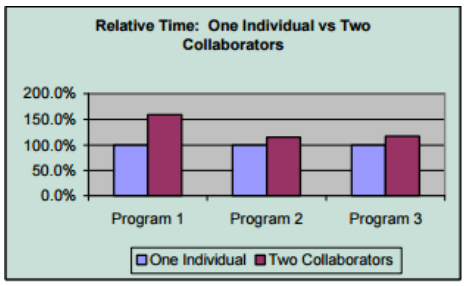
4.1. Porinis programavimas

Pliusai[CW11, BN08]:

- Jei užstringama rašant kodą ir nebežinoma ką daryti toliau, šalia visada yra kitas žmogus susikoncentravęs į tą pačią užduotį. Šis žmogus gali duoti idėjų arba pakeisti vietą prie klaviatūros bei pratęsti darbą. Taip darbas gali vykti greičiau.
- Greičiau pastebimos mažos klaidos, tokios kaip nepadėtas kabliataškis ar nekorektiškai deklaruotas kintamasis. Tai veda prie mažiau laiko skiriamo klaidoms taisyti.
- Geresnė koncentracija. Programuojant poromis maženė tikimybė, kad žmonės nukryps nuo darbų ir pradės užsiiminėti sava veikla susijusia su kompiuteriu (naršys internete).
- Žinių sujungimas. Dažnai poriniame programavime poros žinios būna skirtingai pasiskirsčiusios, todėl programuotojai papildo vienas kito žinias bei darbą padaro greičiau bei kokybiškiau nei darytų tik vienas programuotojas.

Minusai[CW00,CW11]:

- Patirties skirtumai. Patirties skirtumai yra gerai, kai norima išmokyti patirties neturintį programuotoją arba norima supažindinti naują žmogų su projektu, tačiau, jeigu norima validžiai taikyti porinio programavimo praktiką, programuotojų patirtis turi būti panaši, nes kitaip visą darbą daro vienas programuotojas ir judrioji praktika netenka prasmės.
- Dažnai žmonės nesugeba dirbti poromis ir pradeda bendrauti tarpusavyje. Dalis žmonių apskritai nesugeba susikaupti, kai kitas žmogus egzaminuoja jų rašomą kodą.
- Dažnai patyrę programuotojai nesugeba prieiti kompromiso dėl to, kaip reikėtų rašyti kodą (kokį algoritmą naudoti). Programuotojai bando įrodyti, kad jų siūlomas būdas išspręsti problemą yra geriausias. Tokie ginčai dažniausiai būna beprasmiai, nes abu algoritmai būna vienodai geri.
- Nors porinis programavimas yra našesnis, tačiau programuojant individualiai sutaupoma 40-85% laiko (žr. Pav. 2).
- Labai sunku rasti porininką su kuriuo pavyktų efektyviai dirbti.



1 pav. Porinio programavimo našumo atvaizdavimas

4.2. Testavimu grįstas kūrimas

Pliusai[Hi15, WG07]:

- Mažesnė projekto kaina. Nors taikant testavimų grįstą kūrimą parašoma žymiai daugiau kodo, tačiau klaidos randamos ankstyvame kūrimo etape. Microsoft atlikto tyrimo duomenimis taikant testavimų grįstą kūrimą klaidų skaičius sumažinamas 40-90% [Cha09].
- Žymiai lengvesnis klaidų ieškojimas kode, nes visada matai kurioje vietoje klysta testai ir klaidos ieškai būtent toje vietoje.
- Tinkamai naudojamas testavimu grįstas kūrimas veda prie geros architektūros, nes rašant testus prieš sukuriant pačią sistemą iš anksto reikia apgalvoti visus architektūrinius sprendimus bei kaip jie bus įgyvendinti.
- Testai atstoja gyvą dokumentaciją ir padaro kodą geriau suprantamą.

Minusai[WG07]:

- Testavimų grįstas kūrimas užima nemažai laiko. Kūrimas sulėtinamas 15-35%, todėl nedideliems projektams bei projektams, kurie turi būti padaryti labai greitai, šis testavimas nėra tinkamas.
- Testų atnaujinimas užima nemažai laiko, todėl projektas gali netilpti į laiko rėžius (kai jie griežtai apibrėžti).
- Testus turi rašyti tas pats žmogus, kuris kuria projektą, kitu atveju, dažniausiai, testus tenka daug kartų atnaujinti, ko pasakoje mažėja našumas ir kyla projekto kaina.
- Jei reikalavimai yra greitai kintantys, galima sugaišti labai daug laiko rašant testus funkcionalumui, kuris bus greit išmestas/pakeistas.

4.3. Modulių testavimas

Kadangi modulių testavimas dažniausiai taikomas kartu su testavimu grįstu kūrimu, o jis jau apžvelgtas poskyriuje „Testavimu grįstas kūrimas“, toliau bus apžvegti modulių testavimo pliusai ir minusai, kai sistema jau sukurta.

Pliusai[La07]:

- Lengvas regresinis testavimas. Padarius naują pakeitimą nereikia pertestuoti visos sistemos, užtenka perleisti visus automatinius modulių testus ir pamatai ar nebuvo padaryta žalos kitiems moduliams.
- Geresnis kodo suprantamumas.
- Rašant modulių testus egzistuojančiai sistemai galima rasti klaidas, padarytas rašant kodą.

Minusai[Co13,Fa05]:

- Testai gali būti rašomi tam, kad padengtų implementaciją, bet ne reikalavimus.
- Testai, rašomi po implementacijos dažniausiai rašomi taip, kad praeitų, o ne kad ištestuotų tam tikrą modulį.
- Senam, netvarkingai parašytam kodui (ang. Legacy code) šią praktiką įvesti yra ypač sunku ir tai reikalauja žymiai daugiau laiko, todėl dažniausiai ji naudojama pradedant projektą nuo pradžių, o ne prižiūrinčių seną sistemą.

4.3. Išvados

Atsižvelgus į tai, kad mažuma įmonių taiko ir porinį testavimą ir testavimu grįstą kūrima ir šių praktikų naudojimas daro panašų poveikį projektui galima palikti tik vieną gerąją praktiką. Abi praktikos sumažina klaidų skaičių, tačiau taikant testavimų grįstą kūrimą atsiranda daugiau teigiamų dalykų, tokių kaip sumažėjusi dokumentacija bei geresnė architektūra, todėl įmonėms geresnė išeitis būtų palikti testavimų grįstą kūrimą.

Modulių testai yra testavimu grįsto kūrimo dalis. Abi šios praktikos praranda esmę kai sistema yra jau sukurta netaikant šių praktikų ir pagrindinis darbas yra prižiūrėti sistemą. Tokiu atveju šių praktikų taikymas užima labai daug laiko ir yra labai sudėtingas (ypač senam kodui).

Galutinė išvada tyrinėjimo aspektui būtų įtraukti sąlygą, kad jeigu sistema yra kuriama naujai reikėtų naudoti testavimu grįstą kūrimą bei modulių testus, bei atnaujinti testus jei jie yra sukurti ir sistema yra prižiūrima, tačiau jei sistema sukurta nepraktikuojant šių praktkų, jų naudoti ir nereikėtų. Porinio programavimo praktikos reikėtų atsisakyti visais atvejais, nes našumo ir laiko santykis neatsiperka lyginant su dviem individualiais programuotojais.

5. Perėjimo aspektas

7 lentelė. Perėjimo aspekto nagrinėjimas

Judrioji praktika	**Aprašymas**	**Užduotas klausimas**
Dažnos laidos	Kiekviena produkto laida turi būti kiek įmanoma trumpesnė (rekomenduojama nuo 2 iki 4 savaičių). Taip siekiama kaip įmanoma greičiau pristatyti sukurtą funkcionalumą užsakovui.	Ar sistema kuriama iteracijomis? Kiek laiko trunka iteracija? Kas kiek laiko būna leidžiamos (angl. release) naujos programinės įrangos versijos?
Nuolatinė integracija		Ar projektas turi nuolatinę integraciją? Kas kiek laiko integruojate?
Judrusis testavimas	Judrusis testavimas – testavimas, kuris laikosi judriųjų metodikų principų. Tokiam testavimui būdinga: testavimas yra integruotas į kūrimo ciklą, už kokybę atsakinga visa kūrimo komanda, grįžtamasis ryšys iš testavimo veiklos gaunamas labai anksti, visada pasiekiami iškelti kokybės kriterijai.	Ši praktika dubliuojasi su praktikomis priskirtomis konstravimo aspektui, todėl jos įtraukimas į 2 aspektus yra nereikalingas.

Norint atrinkti tik tas komandas, kurios linkusios taikyti perėjimo aspekto judriąsias praktikas buvo atrinktos praktikos, kurių laikosi dauguma komandų. Atskirai pateikiami procentai komandų kuriančių sistemas ir prižiūrinčių sistemas.

Duomenys komandų kuriančių sistemą:

- Ar sistema kuriama iteracijomis? 89.3%

Duomenys komandų prižiūrinčių programinę įrangą:

- Ar sistema kuriama iteracijomis? 86.1%

Atrinkus tik tas komandas, kurios kūrimo/priežiūros procese naudoja iteracijas, buvo iš naujo perskaičiuoti visų perėjimo aspektui priklausančių judriųjų praktikų taikymo duomenys.

8 lentelė. Perėjimo aspekto judriųjų praktikų taikymas

Klausimas:	**Komandos kuriančios sistemas**	**Komados prižiūrinčios sistemas**
Kiek laiko trunka iteracija?	Iki 5 dienų: 21.7% 5-9 dienas: 30.4% 10-15 dienų: 26.1% 23-30 dienų: 4.3% 1-3 mėn.: 17.4%	Iki 5 dienų: 19.4% 5-9 dienas: 25.8% 10-15 dienų: 29% 16-22 dienas: 3.2% 23-30 dienų: 9.7% 1-3 mėn.: 12.9%
Kas kiek laiko būna leidžiamos (angl. release) naujos programinės įrangos versijos?	5-9 dienas: 24% 10-15 dienų: 24% 16-22 dienos: 12% 23-30: 8% 30+: 32%	5-9 dienas: 35.5% 10-15 dienų: 19.4% 16-22 dienos: 3.2% 23-30: 12.9% 30+: 29%
Kas kiek laiko integruojate?	Po kiekvieno pakeitimo: 57.1% Kartą per dieną: 7.1% Kelis kartus per savaitę: 14.3% Kelis kartus per mėnesį: 14.3% Rečiau nei kelis kartus per mėnesį: 7.1%	Po kiekvieno pakeitimo: 52.2% Kelis kartus per savaitę: 30.2% Kelis kartus per mėnesį: 11.5% Rečiau nei kelis kartus per mėnesį: 6.2%

Perėjimo aspekte dauguma komandų visas judriąsias praktikas atlieka korektiškai.

Ir komandos kuriančios ir prižiūrinčios sistemas turi panašaus ilgio iteracijas. Gerosios praktikos „dažnos laidos“ apibrėžimą atitinka daugiau nei 80% komandų, kurių iteracijos trunka mažiau nei 30 dienų (~4 savaites). Apie 70% komandų po kiekvienos iteracijos išleidžia naują programinės įrangos versiją į produkcijos aplinką.

Virš 50% kuriančių bei prižiūrinčių sistemas komandų, naudojančių nuolatinės integracijos praktiką, integruoja padarytą darbą po kiekvieno pakeitimo. Tai atitinka judriojo kūrimo modelio reikalavimus.

5.1. Išvados

Visas perėjimo aspektui priklausančias praktikas įmonės naudoja savo kasdieniniame darbe, todėl jos visos turėtų būti paliktos. Šis aspektas yra sudarytas korektiškai ir atitinka realų komandų darbą.

6. Valdymo aspektas

9 lentelė. Valdymo aspekto nagrinėjimas

Judrioji praktika	**Aprašymas**	**Užduotas klausimas**
Darbų sąrašas (angl. Product Backlog) –	Visko kas gali būti reikalinga produktui sąrašas ir vienintelis produkto reikalavimų šaltinis. Šis sąrašas nuolat keičiasi – ankstyvoji jo versija remiasi tik pradžioje žinomais reikalavimais vėliau keičiasi drauge su produktu ir aplinka, kurioje jis bus naudojamas. Už produkto darbų sąrašo valdymą yra atsakingas produkto savininkas.	Ar turite produkto darbų sąrašus (angl. Product backlog), kuriuose darbai surašyti su prioritetais? Kuriuo metu surenkami reikalavimai sistemai?
Planavimo žaidimas	Verslo ir techninių prioritetų nustatymas ir reikalavimų priskyrimas iteracijoms.	Ar planuodami užduotis žaidžiate planavimo žaidimus (pvz scrm pokeris)?
Vizuali ir prieinama projekto informacija (angl. Information radiotor).	Projekto informacija turi būti pavaizduota paprastai ir prieinama visiems komandos nariams. Tokią informaciją gali sudaryti: Esamos-planuotos situacijos diagrama (angl. Gantt chart) – tai grafinis esamos ir planuotos situacijos atvaizdavimas. Užduočių lenta (angl. Task board) – tai iteracijos darbų sąrašas su aktualia kiekvienos užduoties būsena	Ar turite produkto darbų sąrašus (angl. Product backlog), kuriuose darbai surašyti su prioritetais? Darbų valdymo priemonės: Scrum board iteracijai Kanban board Paprastas darbų sąrašas

Judrusis projektų darbų apimties vertinimas		Ar turite produkto darbų sąrašus (angl. Product backlog), kuriuose darbai surašyti su prioritetais?
Komandos susirinkimai	Komandos susirinkimai vyksta su aiškiu tikslu, aiškiais dalyviais ir paprastai numatyta maksimalia trukme. Pagrindiniai susirinkimų tikslai – sudaryti žmonėms sąlygas efektyviam bendradarbiavimui, vesti prie „žinių suvisuomeninimo„ (visi žino tikslią projekto būseną) ir skatinti saviorganizuojančią komandos struktūrą. Turi būti vykdomi tokie susirinkimai: trumpi (tradiciškai 15 minučių) kasdieniniai komandos susirinkimai, sprinto planavimo susirinkimai, sprinto peržiūros susirinkimai, sprinto retrospektyvos susirinkimai.	Kas kiek laiko organizuojami komandos susitikimai? Kokius susirinkimus praktikuojate? Trumpi (tradiciškai 15 minučių) kasdieniai komandos susirinkimai Savaitiniai susitikimai Iteracijos planavimo susirinkimai Iteracijos peržiūros susitikimai Iteracijos retrospektyvos susirinkimai Kita

Norint atrinkti tik tas komandas, kurios linkusios taikyti valdymo aspekto judriąsias praktikas buvo atrinktos praktikos, kurių laikosi dauguma komandų. Atskirai pateikiami procentai komandų kuriančių sistemas ir prižiūrinčių sistemas.

Duomenys komandų kuriančių naują sistemą:

- Ar turite produkto darbų sąrašus (angl. Product backlog), kuriuose darbai surašyti su prioritetais? 85.7%

Komanos prižiūrinčios sistemą:

- Ar turite produkto darbų sąrašus (angl. Product backlog), kuriuose darbai surašyti su prioritetais? 83.3%

Atrinkus tik tas komandas, kurios taiko šią judriąją praktiką buvo iš naujo perskaičiuoti visų valdymo aspektui priklausančių judriųjų praktikų taikymo duomenys.

10 lentelė. Valdymo aspekto judriųjų praktikų taikymas

Klausimas:	**Komandos kuriančios sistemas**	**Komados prižiūrinčios sistemas**
Kuriuo metu surenkami reikalavimai sistemai?	Projekto pradžioje ir gali keistis bei atsirasti naujų vystymo eigoje: 62.5% Projekto eigoje palaipsniui: 33.3% Projekto pradžioje ir negali keistis: 4.2%	Projekto pradžioje ir gali keistis bei atsirasti naujų vystymo eigoje: 73.3% Projekto eigoje palaipsniui: 20% Projekto pradžioje ir negali keistis: 6.7%
Ar planuodami užduotis žaidžiate planavimo žaidimus (pvz. Scrum pokeris)?	Taip: 23.4% Ne 76.6%	Taip: 14.7% Ne 85.3%
Darbų valdymo priemonės:	Scrum board iteracijai: 37.5% Kanban board: 23.8% Paprastas darbų sąrašas: 37.5% Kita: 33.3%	Scrum board iteracijai: 40% Kanban board: 36.7% Paprastas darbų sąrašas: 30% Kita: 40%
Kas kiek laiko organizuojami komandos susitikimai?	Kelis kartus per dieną: 8.3% Kartą per dieną: 29.2% Daugiau nei 2 kartus per savaitę: 20.8% Mažiau arba 2 kartus per savaitę: 20.8% Kartą per 2 savaites: 8.3% Rečiau nei kartą per 2 savaites: 4.2%	Kelis kartus per dieną: 6.7% Kartą per dieną: 30% Daugiau nei 2 kartus per savaitę: 20% Mažiau arba 2 kartus per savaitę: 26.7% Kartą per 2 savaites: 6.7% Rečiau nei kartą per 2 savaites: 3.3%

	Visai neorganizuojami: 8.3%	Visai neorganizuojami: 6.7%
Kokius susirinkimus praktikuojate?	Trumpi (tradiciškai 15 minučių) kasdieniai komandos susirinkimai: 54.1% Savaitiniai susitikimai: 41.6% Iteracijos planavimo susirinkimai: 58.3% Iteracijos peržiūros susitikimai: 45.8% Iteracijos retrospektyvos susirinkimai: 37.5% Kita: 20.8%	Trumpi (tradiciškai 15 minučių) kasdieniai komandos susirinkimai: 63.3% Savaitiniai susitikimai: 60% Iteracijos planavimo susirinkimai: 48.7% Iteracijos peržiūros susitikimai: 40% Iteracijos retrospektyvos susirinkimai: 23.3% Kita: 23.3%

Geroji praktika „Darbų sąrašas“ yra taikoma visose atrinktose komandose. Daugumoje komandų reikalavimai yra kintantys ir nuolat atnaujinami. Darbų sarašai yra vizualūs ir prieinami daugumai komandų.

Planavimo žaidimai yra nežaidžiami daugumoje komandų, todėl plačiau bus aptarti skyriuje „Planavimas“.

Komandų susitikimus praktikuoja dauguma komandų. Mažiau praktikuojami retrospektyvos susitikimai, todėl jie bus aptarti poskyriuje „Iteracijos retrospektyvos susitikimai“.

6.1. Planavimas

Aprašomas konkretaus planavimo žaidimo, planavimo (arba Scrum) pokerio, pliusai ir minusai.

Pliusai[Po14]:

- Susitikimo metu planuojant kitos iteracijos užduočių laikus gerai naudoti kortas vietoje žodinio pasisakymo, nes pasisakius pirmąjam žmogui kitų atsakymai gali būti paveikti jo atskymo.
- Planavimo metu visoms užduotims priskiriami preliminarūs laikai, todėl iteracijos metu programuotojai iškart turi nuovoką, kiek maždaug laiko ši užduotis truks ir neapsiema užduoties, jei nespės jos padaryti.
- Diskusijų metu iškeliamos problemos su kuriomis programuotojas gali susidurti ir trumpai aptariami sprendimai, todėl programuotojams lengviau vykdyti užduotį ją gavus.

- Planavimo žaidimai nepaveikia neigiamai nustatytų prioritetų, nes užduotys aptariamos retrospektyvos metu imamos pagal prioritetus.

Minusai[Po14]:

- Dažniausiai likus paskutiniems taškams planuojant sekantį sprintą renkama ne ta užduotis, kuri yra aukščiausia pagal prioritetus, bet ta, kuri atitinka spėjamo laiko likutį.
- Išklausomos tik tų programuotojų nuomonės, kurių nurodomi laikai yra ekstremumai (didžiausia arba mažiausia reikšmė), nors gali būti taip pat svarbi ir kitų žmonių nuomonė (pavyzdžiui, gal kažkas jau darė panašią užduotį ir žino, kiek maždaug laiko reikės užduoties įvykdymui).
- Negali pasirinkti bet kokių laiko reikšmių. Reikšmės yra iš karto nustatytos (fibonačio sekos skaičiai iki 21, paskui 100).
- Jei komanda nesusigyvenus ir nepatyrusi gali skirti laiko taškus neatitinkančius realbės. Tokiu atveju iteracijos gale gali nebenūti užduočių arba jų gali būti per daug.

6.2. Iteracijos retrospektyvos susitikimai

Pliusai[GL13, Lar06]:

- Retrospektyvos leidžia komandai pačiai nuspręsti, kokios projekto kūrimo proceso problemos yra didžiausios, apsitarti, kokias praktikas reikėtų taikyti, kad jų išvengti.
- Komanda yra motyvuota keisti blogas praktikas į geras, nes tai padės išspręsti jų pačių iškeltas problemas.
- Retrospektyvų sisitikimuose žmonės laisvai gali kalbėti apie savo nusivylimus ir problemas kuriant sistemą. Juos išklauso visi komandos nariai ir pasiūlo, kaip tai išspręsti.
- Po retrospektyvų žmonės taiko naujas praktikas savo darbe gerindami kūrimo procesą. Tai veda prie nuolatinio tobulėjimo (angl. *Continuous Improvement*).
- Gerinant kūrimo procesą komanda dirba našiau bei efektyviau. Tai veda prie dažnesnio sistemos perdavimo užsakovui.

Minusai[Mc13]:

- Komandos, nors ir išdiskutavusios problemas bei suradusios, kaip būtų galima pagerinti procesą, dažnai netaiko naujų praktikų kūrimo procese, todėl retrospektyvos būna tik laiko švaistymas.
- Neturint gero lyderio, kuris vestų retrospektyvą, susitikimai tampa vieta, kur žmonės diskutuoja, ką kitos komandos, o ne pati komanda turėtų daryti bei apie vienkartines problemas, kurias išsprendus kūrimo procesas nepagerėtų arba apie tai, ko nereikėtų arba nepavyks pakeisti sekančioje iteracijoje.

- Komandos, kurios tik pradeda taikyti retrospektyvų susitikimus ne visada supranta to paskirties ir taiko retrospektyvas, kad būtų judresni. Tokiu atveju retrospektyvos praranda esmę, nes net jei visi aptaria problemas, nieks neplanuoja toliau kažką keisti darbe.
- Senai suformuotoms komandoms labai sunku keisti įpročius programinės įrangos kūrimo procese.

6.3. Išvados

Planavimo žaidimai yra greitas ir efektyvus būdas suplanuoti kokius uždavinius reikėtų daryti sekančioje iteracijoje, tačiau kolkas nėra sukurto kiekvienai komandai tinkančio ir minusų neturinčio planavimo žaidimo, todėl kiekviena komanda turėtų pakoreguoti planavimo žaidimus pagal savo poreikius, pavyzdžiui, žaidžiant Scrum pokerį reikėtų naudoti ne tam iš anksto sukurtas kortas, o kiekvienas komandos narys ant lapelio turėtų užrašyti tokį skaičių laiko taškų, kiek, jam atrodo, užtruks padaryti uždavinį (reikėtų atsisakyti iš anksto numatytų reikšmių).

Iteracijos retrospektyvos susitikimai taip pat yra gera praktika vedanti prie nuolatos gerėjančio kūrimo proceso, tačiau norint ją pradėti taikyti komandoje, reikia supažindinti visus komandos narius su šios judriosios praktikos pliusais ir paskirti atsakingą žmogų, pavyzdžiui, jei komandoje yra, scrum meistrą, kuris vestų šiuos susirinkimus ir tikrintų ar komanda keičia savo įpročius iteracijos eigoje.

Galutinė išvada - visos perėjimo aspektui priskirtos praktikos yra geros ir tinka tiek kuriančioms, tiek prižiūrinčioms sistemas komandoms.

7. Kultūros aspektas

11 lentelė. Kultūros aspekto nagrinėjimas

Judrioji praktika	**Aprašymas**	**Užduotas klausimas**
Užsakovo įtraukimas į kūrimo procesą		Per kiek laiko užsakovas atsako į patikslinančius klausimus? Per kiek laiko užsakovas patvirtina galutines versijas (pvz. reikalavimų, priėmimo kriterijų, dizainų)?
Jokių viršvalandžių	Ši taisyklė sako tai, kad darbuotojai neturi dirbti pavargę. 40-ies darbo valandų savaitė	Ar tenka dirbti viršvalandžius?
Fizinė aplinka	Komanda turi dirbti aplinkoje, kurioje atsiskleistų geriausios judriųjų metodikų savybės. Tokios aplinkos pagrindas – tai darbas bendroje erdvėje, kur komandos nariai galėtų bendrauti nevaržomai.	Ar visa komanda (išskyrus "product owner") dirba viename kabinete ir gali nevaržomai bendrauti?
Saviorganizuojančios komandos	Komandai suteikta teisė pačiai priimti sprendimus, be vadovų patvirtinimo. Tokiose komandose dažniausiai rolės nėra griežtai apibrėžtos, kai komanda gauna užduotį, už užduoties įvykdymą yra atsakinga visa komanda, ne konkretus komandos narys.	Ar rolės jūsų komandoje yra griežtai apibrėžtos? Kokių rolių atstovai kuria/prižiūri jūsų projektą?

Norint atrinkti tik tas komandas, kurios linkusios taikyti valdymo aspekto judriąsias praktikas buvo atrinktos esminės praktikos, kurių laikosi dauguma komandų. Atskirai pateikiami procentai komandų kuriančių sistemas ir prižiūrinčių sistemas.

Duomenys komandų kuriančių sistemą:

- Darbuotojai niekada nedirba viršvalandžių arba dirba su retomis išimtimis: 78.8%
- Visa komanda dirba vienoje patalpoje: 71.4%

Komandos prižiūrinčių sistemas:

- Darbuotojai niekada nedirba viršvalandžių arba dirba su retomis išimtimis: 77.75%
- Visa komanda dirba vienoje patalpoje: 75%

Atrinkus tik tas komandas, kurios taiko abi judriąsias praktikas buvo iš naujo perskaičiuoti visų kultūros aspektui priklausančių judriųjų praktikų taikymo duomenys.

12 lentelė. Kultūros aspekto judriųjų praktikų taikymas

Klausimas:	**Komandos kuriančios sistemas**	**Komados prižiūrinčios sistemas**
Per kiek laiko užsakovas atsako į patikslinančius klausimus?	Per 0.5h: 16.7% Per 1-2h: 16.7% Per tą pačią dieną: 38.9% Kitą dieną: 11.1% Vėliau nei kitą dieną: 16.7%	Per 0.5h: 29.2% Per 1-2h: 16.7% Per tą pačią dieną: 37.5% Kitą dieną: 16.7%
Per kiek laiko užsakovas patvirtina galutines versijas (pvz. reikalavimų, priėmimo kriterijų, dizainų)?	Per 1-2h: 22.1% Per tą pačią dieną: 16.7% Kitą dieną: 38.9% Vėliau nei kitą dieną: 23.3%	Per 1-2h: 20.8 Per tą pačią dieną: 25% Kitą dieną: 33.3% Vėliau nei kitą dieną: 20.8%
Ar rolės jūsų komandoje yra griežtai apibrėžtos?	Taip: 42.9% Ne: 57.1%	Taip: 45.6% Ne: 56.3%
Kokių rolių atstovai kuria/prižiūri jūsų projektą?	Programuotojai: 100% Testuotojai: 61.1% Projekto vadovas: 77.8% Analitikai: 38.9% Product owner: 44.4%	Programuotojai: 100% Testuotojai: 58.3% Projekto vadovas: 79.2% Analitikai: 41.7% Product owner: 29.2%

Daugiau nei 70% ir kuriančių, ir prižiūrinčių komandų užsakovas atsako į patikslinančius klausimus tą pačią dieną ir per 2 dienas patvirtina galutines versijas. Tai yra pakankamas skaičius, kad būtų galima judriąją praktiką „Užsakovo įtraukimas į kūrimo procesą“ laikyti tinkamai įtraukta į judrumo proceso vertinimo modelį.

Daugumoje komandų rolės nėra griežtai apibrėžtos. Tai reiškia, kad komandose programuotojai gali atlikti ir testuotojo ar architekto rolę. Ši praktika atitinka praktikos „Saviorganizuojančios komandos“ aprašymą ir turi būti įtraukta į proceso judrumo vertinimo modelį.

7.1. Išvados

Visas kultūros aspektui priklausančias praktikas komandos naudoja savo kasdieniniame darbe, todėl jos visos turėtų būti paliktos judrumo vertinimo modelyje. Šis aspektas yra sudarytas korektiškai ir atitinka realų komandų darbą.

Išvados ir rezultatai

Bakalaurinio darbo metu ištirta, kaip komandos taiko proceso judrumo modeliui priskirtas judriąsias praktikas ir ar tie patys praktikų rinkiniai tinka ir programų kūrimo ir priežiūros procesuose. Išsiaiškinta, kad ne visas judriąsias praktikas, įtrauktas į proceso judrumo modelio aspektų dimensiją komandos taiko savo darbe bei ne visos praktikos vienodai naudojamos komandose, kuriančiose ir komandose prižiūrinčiose projektus.

Šis darbas rėmėsi 53 Lietuvoje dirbančių komandų atsakymais į klausimus apie taikomas praktikas komadoje, kurioje dirba atsakiusieji. Buvo tiriama, ar komanda taiko daugumą konkrečiai aspektų dimencijai priklausančių praktikų. Jei dauguma šių komandų taiko tas pačias praktikas, tačiau netaiko kažkurios praktikos, buvo daromas tyrimas, kodėl ta praktika netaikoma bei pateikiami pagrindiniai judriosios praktikos pliusai ir minusai. Dauguma komandų netaikė šių aspektų dimensijai priskirtų judriųjų praktikų:

- modeliavimas ir prototipavimas;
- porinis programavimas;
- testavimu grįstas kūrimas;
- modulių testavimas;
- planavimas;
- iteracijos retrospektyvos susitikimai.

Šio darbo metu buvo prieita prie išvados, kad įmonėms nevertėtų laikytis judriosios praktikos „Porinis programavimas“. Ši praktika įmonėms yra labai brangi, nes vietoje vieno programuotojo tą patį darbą daro du programuotojai, o jų darbas būna tik 1.3 karto efektyvesnis nei vieno programuotojo darbas. Didžiausią šios praktikos pliusą - greitai pastebimas smulkias klaidas padengia kita to pačio aspekto praktika - „testavimu grįstas kūrimas“.

Kitas šio darbo tikslas buvo patikrinti, ar tos pačios judriosios praktikos tinka ir kūrimo ir priežiūros procesuose. Buvo prieita prie išvados, kad komandoms prižiūrinčioms sistemą netinka judrioji praktika „Modeliavimas ir prototipavimas“, nes jie neturi kam kurti prototipų. Komandoms, prižiūrinčioms sistemą, netaikančioms praktikų „Testavimu grįstas kūrimas“ bei „Modulių testavimas“ nereikėtų ir pradėti taikyti šių judriųjų praktikų, nes tai būtų labai brangu, užimtų labai daug laiko ir nevisada pavyktų padaryti (ypač senam kodui). Tokioms komandoms geriau taikyti integracijų testus.

Paredaguotas judrumo vertinimo modelis (žr. lent. 13) turėtų turėti atskiras aspektų dimencijai priskirtas judriąsias praktikas skirtas kūrimo bei priežiūros procesams.

13 lentelė. Pakoreguotas kokybinis judriųjų praktikų ir aspektų gretinimas

Aspektas	**Judriosios praktikos kūrimo procesui**	**Judriosios praktikos sistemos priežiūros procesui**
Tyrinėjimo	Vartotojų pasakojimai	Vartotojų pasakojimai
	Evoliuciniai reikalavimai	Evoliuciniai reikalavimai
	Modeliavimas ir prototipavimas	-
Konstravimo	Kodo pertvarkymas	Kodo pertvarkymas
	Modulių testai	Modulių testai*
	Bendra kodo nuosavybė	Bendra kodo nuosavybė
	Kodo standartai	Kodo standartai
	Testavimu grįstas kūrimas	Testavimu grįstas kūrimas*
Perėjimo	Dažnos laidos	Dažnos laidos
	Nuolatinė integracija	Nuolatinė integracija
	Judrusis testavimas	Judrusis testavimas
Valdymo	Darbų sąrašas	Darbų sąrašas
	Planavimo žaidimas	Planavimo žaidimas
	Vizuali ir prieinama projekto informacija	Vizuali ir prieinama projekto informacija
	Judrusis projektų darbų apimties vertinimas	Judrusis projektų darbų apimties vertinimas
	Komandos susirinkimai	Komandos susirinkimai
Kultūros	Užsakovo įtraukimas į kūrimo procesą	Užsakovo įtraukimas į kūrimo procesą
	Jokių viršvalandžių	Jokių viršvalandžių

	Fizinė aplinka	Fizinė aplinka
	Saviorganizuojančios komandos	Saviorganizuojančios komandos

*Taikoma tik tuo atveju, jei komanda šią praktiką taikė nuo kūrimo pradžios.

Šaltiniai

[Ma15] Nerijus Mazėtis, Proceso judrumo vertinimo modelis, 2015.

[OD14] Ö. Özcan Top and O. Demirörs, Assessing Software Agility: An Exploratory Case Study.

[AM01] Agile Manifesto, 2001. Prieiga per internetą: www.agilemanifesto.org

[SEI07] Carnegie Mellon University. Capability Maturity Model Integration (CMMI) Version 1.2, 2007

[Fa05] Michael Feathers, Working Effectively with Legacy Code, 2005

[La07] Jeff Langr, Doing TDD Well, 2007

[CW00] Alistair Cockburn, Laurie Williams, The Costs and Benefits of Pair Programming, 2000

[GL13] Luis Gonçalves and Ben Linders, Getting Value out of Agile Retrospectives, 2013

[Lar06] Diana Larsen, Agile Retrospectives Making Good Teams Great, 2006

[Cha09] Janie Chang, Exploding Software-Engineering Myths, 2009

[TDD2] Simon Hill, The pros and cons of Test-Driven Development, 2015

[Co13] James O Coplien, Why Most Unit Testing is Waste, 2013

[Po14] Stefano Porro, „Story points“ or „How Agile creates democracy in estimates“ – Part 2, 2014

[Mc13] Keith McMillan, Retrospective, 2013

[Ne04] Petra Neumann, Prototyping, 2004

[MSP13] Methods for Software Prototyping, 2013

[CW11] Alistair Cockburn, Laurie Williams, The Costs and Benefits of Pair Programming, 2011

[BN08] Andrew Begel, Nachiappan Nagappan, Pair Programming: What's in it for Me? 2008

[WG07] Hans Wasmus, Hans-Gerhard Gross, Evaluation of Test-Driven Development, 2007

ISO04 ISO/IEC 33004: Information technology - Process Assessment - Requirements for process reference, process assessment and maturity models