



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ TECHNOLOGIJŲ KATEDRA

Evaldas Rekašius

DARBO METODOLOGIJŲ PALYGINIMAS BEI EFEKTYVUMO TYRIMAS

**COMPARISON OF WORK METHODOLOGIES AND ANALYSIS OF
EFFECTIVENESS**

Baigiamasis magistro darbas

Inžinerinės informatikos studijų programa, valstybinis kodas 621E14004

Duomenų gavybos technologijų specializacija

Informatikos inžinerijos studijų kryptis

Vilnius, 2013

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ TECHNOLOGIJŲ KATEDRA

TVIRTINU
Katedros vedėjas

(Parašas)

(Vardas, Pavardė)

(Data)

Evaldas Rekašius

DARBO METODOLOGIJŲ PALYGINIMAS BEI EFEKTYVUMO TYRIMAS
COMPARISON OF WORK METHODOLOGIES AND ANALYSIS OF
EFFECTIVENESS

Baigiamasis magistro darbas

Inžinerinės informatikos studijų programa, valstybinis kodas 621E14004

Duomenų gavybos technologijų specializacija

Informatikos inžinerijos studijų kryptis

Vadovas

(Moksl. laipsnis/pedag. vardas, vardas, pavardė)

(Parašas)

(Data)

Lietuvių kalbos konsultantas

(Moksl. laipsnis/pedag. vardas, vardas, pavardė)

(Parašas)

(Data)

Vilnius, 2013

Vilniaus Gedimino technikos universiteto
egzaminų, sesijų ir baigiamųjų darbų rengimo bei
gynimo organizavimo tvarkos aprašo
2011-2012 m. m.

1 priedas

(Baigiamojo darbo sąžiningumo deklaracijos forma)

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Evaldas Rekašius, 20074906

(Studento vardas ir pavardė, studento pažymėjimo Nr.)

Fundamentinių mokslų fakultetas

(Fakultetas)

Informacinės technologijos, DGTfm-11

(Studijų programa, akademinė grupė)

**BAIGIAMOJO DARBO (PROJEKTO)
SĄŽININGUMO DEKLARACIJA**

2013 m. gegužės 29 d.

Patvirtinu, kad mano baigiamasis darbas tema „Darbo metodologijų palyginimas bei efektyvumo tyrimas“ patvirtintas 2012 m. spalio 24 d. dekanų potvarkiu Nr. 375fm, yra savarankiškai parašytas. Šiame darbe pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose.

Mano darbo vadovas prof. habil. dr. Genadijus Kulvietis.

Kitų asmenų indėlio į parengtą baigiamąjį darbą nėra. Jokių įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs (-usi).



(Parašas)

Evaldas Rekašius

(Vardas ir pavardė)

Vilniaus Gedimino technikos universitetas
Fundamentinių mokslų fakultetas
Informacinių technologijų katedra

ISBN ISSN
Egz. sk.
Data

Antrosios pakopos studijų **Informacinių technologijų** programos baigiamasis darbas

Pavadinimas: **Darbo metodologijų palyginimas bei efektyvumo tyrimas**

Autorius **Evaldas Rekašius**

Vadovas **prof. habil. dr. Genadijus Kulvietis**

Kalba

☒

lietuvių

☐

užsienio

Anotacija

Baigiamojo darbo tikslas – atlikti dviejų darbo metodologijų palyginimą bei efektyvumo tyrimą.

Šiame darbe buvo atlikti *Scrum* ir *Kanban* metodologijų efektyvumo tyrimai ir abiejų metodologijų efektyvumo palyginamoji analizė. Analizuojant metodologijų efektyvumą buvo įvertinta, kuri metodologija naudingesnė.

Prieš pradedant tirti darbo metodologijų efektyvumą buvo išanalizuotos dvi darbo metodologijos: *Scrum* ir *Kanban*. Analizuojant abi metodologijas buvo įvertinami vienos ir kitos metodologijos pranašumai bei trūkumai.

Darbą sudaro 5 dalys: įvadas, *scrum* metodologijos analizė, privalumai ir trūkumai, *kanban* metodologijos analizė, privalumai ir trūkumai, darbo efektyvumo tyrimas, naudojant *Scrum* ir *Kanban* metodologijas, išvados bei literatūros sąrašas.

Darbo apimtis – 54psl. su 1 priedu, 35pav. ir 21 literatūros šaltinis.

Prasminiai žodžiai:

Scrum, Kanban, Sprint, Sprint planning, Kanban board

Vilnius Gediminas Technical University
Fundamental Sciences faculty
Information Technologies department

ISBN ISSN
Copies No.
Date

Master Degree Studies **Information Technologies** study programme Final thesis

Title: **Comparison of work methodologies and analysis of effectiveness**

Author **Evaldas Rekašius**

Academic supervisor **Prof Dr Habil Genadijus Kulvietis**

Thesis language

☒ Lithuanian

☐ Foreign (English)

Annotation

The point of this work is to perform comparison of work methodologies and analysis of effectiveness.

There was performed *Scrum* and *Kanban* methodologies analysis of effectiveness at this work and comparative analysis of both methodologies effectiveness. During the analysis of effectiveness was evaluated which is more useful.

There were analyzed two work methodologies: *Scrum* and *Kanban* before starting to investigate the effectiveness of work methodologies. Advantages and disadvantages were evaluated for both methodologies during the analysis of methodologies.

The work consists of 5 parts: introduction, analysis of *Scrum* methodology, analysis of *Kanban* methodology, analysis of work effectiveness using *Scrum* and *Kanban* methodologies, conclusions and references.

The size of the work: 54 pages with 1 appendix, 35 figures and 21 references.

Keywords:

Scrum, Kanban, Sprint, Sprint planning, Kanban board

Turinys

Paveikslų sąrašas	7
Santrumpų ir terminų žodynėlis	9
Įvadas.....	10
1. <i>Scrum</i> metodologijos analizė, privalumai ir trūkumai.....	12
1.1. Įvadas.....	12
1.2. <i>Scrum</i> metodologijos analizė.....	12
1.3. <i>Scrum</i> metodologijos trūkumai.....	22
1.4. <i>Scrum</i> metodologijos privalumai.....	22
1.5. Išvados	23
2. <i>Kanban</i> metodologijos analizė, privalumai ir trūkumai	24
2.1. Įvadas.....	24
2.2. <i>Kanban</i> metodologijos analizė	24
2.3. <i>Kanban</i> metodologijos privalumai	36
2.4. <i>Kanban</i> metodologijos trūkumai	36
2.5. Išvados	37
3. Darbo efektyvumo tyrimas, naudojant <i>Scrum</i> ir <i>Kanban</i> metodologijas	38
3.1. Įvadas.....	38
3.2. Darbo efektyvumas naudojant <i>Scrum</i>	38
3.3. Darbo efektyvumas naudojant <i>Kanban</i>	42
3.4. Darbo efektyvumo tyrimo analizė	46
3.5. Išvados	50
Išvados ir rekomendacijos	51
Literatūra	52
Priedas	54

Paveikslų sąrašas

1 pav.	<i>Scrum</i> metodo procesas	13
2 pav.	Iteracijos progreso diagrama	15
3 pav.	Naujos versijos išleidimo diagrama.....	16
4 pav.	Nepriklausomos komandos	16
5 pav.	1-ojo lygio skalinis modelis.....	17
6 pav.	2-ojo lygio skalinis modelis.....	18
7 pav.	Komandos kaip „Klientai“	19
8 pav.	<i>Scrum</i> susirinkimų lygmenys.....	21
9 pav.	<i>Kanban</i> lenta.....	25
10 pav.	Pagrindiniai darbo procesai	26
11 pav.	<i>Kanban</i> lenta.....	27
12 pav.	<i>Kanban</i> lenta.....	30
13 pav.	Pagal limitą paskiriamos užduotys	30
14 pav.	Programuojamos ir toliau skirstomos užduotys	31
15 pav.	Atlikta užduotis atiduodama diegimui.....	31
16 pav.	Programuojamos ir diegiamos užduotys.....	32
17 pav.	Susiduriama su problemomis diegiant užduotį.....	32
18 pav.	Pasiektas maksimalus programavimo etapo limitas	33
19 pav.	Programuotojai padeda diegimo inžinieriams	33
20 pav.	Programuotojai padeda diegimo inžinieriams	34
21 pav.	Programuotojai padeda diegimo inžinieriams	34
22 pav.	Visa komanda stengiasi išspręsti diegimo problemą.....	35
23 pav.	Sėkminga proceso eiga išsprendus problemas.....	35
24 pav.	Atlikti istorijos taškai per iteracijas naudojant <i>Scrum</i>	39
25 pav.	Vidurkio ir atliktų istorijos taškų palyginimas naudojant <i>Scrum</i>	40
26 pav.	Darbo laikas, praleistas įgyvendinant užduotis, naudojant <i>Scrum</i>	41
27 pav.	Darbo laikas, praleistas vienam istorijos taškui įgyvendinti, naudojant <i>Scrum</i>	42
28 pav.	Atlikti istorijos taškai per iteracijas naudojant <i>Kanban</i>	43
29 pav.	Vidurkio ir atliktų istorijos taškų palyginimas naudojant <i>Kanban</i>	44
30 pav.	Darbo laikas, praleistas įgyvendinant užduotis, naudojant <i>Kanban</i>	45

31 pav.	Darbo laikas, praleistas vienam istorijos taškui įgyvendinti, naudojant <i>Kanban</i>	46
32 pav.	Atliktų istorijos taškų palyginimas naudojant abi metodologijas.....	47
33 pav.	Atliktų istorijos taškų per iteraciją palyginimas naudojant abi metodologijas.....	48
34 pav.	Darbo laiko per visą laikotarpį palyginimas naudojant abi metodologijas	49
35 pav.	Darbo laiko palyginimas vienam istorijos taškui įgyvendinti	50

Santrumpų ir terminų žodynėlis

Living backlog – darbų sąrašas

Sprint – iteracija

Sprint planning – iteracijos planavimas

Scrum master – scrum vedlys

Retrospective meeting – retrospektyvos susirinkimas

Sprint burndown chart – iteracijos progreso diagrama

Release burndown chart – versijos išleidimo diagrama

Story points – istoriniai taškai

Impediments – kliūtys

WIP limits – angl. Work In Progress Limits – neužbaigtų darbų apribojimas

Kanban board – kanban lenta

Bottlenecks – sangrūdos

New feature – nauja užduotis

Įvadas

Aktualumas. Nepaisant to, jog kone visos informacinių technologijų įmonės, kuriančios programinę įrangą, naudoja įvairias metodologijas, skirtas darbo procesui optimizuoti, dauguma iš jų vis dar ieško tokios metodologijos, kuri neturėtų jokių trūkumų ir atitiktų visus poreikius. Tiek programinės įrangos kūrėjai, tiek klientai ieško būdų, kaip būtų galima sumažinti kuriamos programinės įrangos kainą, kad nenukentėtų jos kokybė, kūrimo procesas neužsitęstų per ilgai ir būtų patenkinti visi klientų poreikiai bei reikalavimai.

Yra daugybė veiksnių, kurie nulemia sėkmingą ar nesėkmingą programinės įrangos kūrimo procesą. Skirtingos laiko juostos, darbuotojų kompetencija, kalbos barjeras, gebėjimas greitai reaguoti į netikėtai iškilusias problemas – šie ir daugelis kitų veiksnių gali nulemti visą kūrimo eigą. Dėl visų šių ir kitų veiksnių nesuderinamumo yra gaišamas brangus laikas bei gali padidėti produkto kaina. Tik sugebant sėkmingai suderinti visus ar bent jau didžiąją dalį veiksnių, galima tikėtis sėkmingos programinės įrangos kūrimo eigos.

Mokslinis naujumas. Sparčiai tobulėjant informacinėms technologijoms, atsiranda vis daugiau naujovių, nepaisant jų, yra dalis sričių, kuriose dirbama pagal nusistovėjusias praktikas, tačiau programinės įrangos kūrimo procesuose nusistovėjusios praktikos nėra.

Vis daugiau kompanijų, kurdamos programinę įrangą, pradeda naudoti iteracinius programinės įrangos kūrimo procesus, lanksčiojo programinės įrangos kūrimo projektų valdymo metodus (angl. *agile*) ir įsitikina jų nauda. *Scrum* – ganėtinai populiarus lanksčiojo programavimo projektų valdymo metodas, neretai pasirenkamas dėl to, kad, pradėjusios naudoti šį metodą, komandos pradeda daugiau bendrauti ir bendradarbiauti, klientų reikalavimai gali būti valdomi lanksčiau per trumpą laikotarpį ir daugelio kitų savybių. *Kanban* – kitas, mažiau populiarus lanksčiojo programavimo projektų valdymo metodas, kuris, ko gero, yra rečiau naudojamas būtent dėl mažesnio populiarumo, tačiau yra ne ką prastesnis, o gal ir pranašesnis nei minėtasis *Scrum* metodas.

Darbo tikslas – atlikti darbo metodologijų palyginimą bei efektyvumo tyrimą programinės įrangos kūrimo įmonėje.

Uždaviniai tikslui pasiekti:

- Atlikti *Scrum* ir *Kanban* metodologijų analizę
- Atlikti *Scrum* ir *Kanban* metodologijų efektyvumo tyrimą
- Palyginti išanalizuotas metodologijas ir efektyvumo tyrimo rezultatus bei pateikti išvadas.

Praktinė darbo vertė. Atliktos *Scrum* ir *Kanban* metodologijų analizės, efektyvumo tyrimo naudojant tiek vieną, tiek kitą metodologiją bei lyginamosios analizės rezultatai gali būti pritaikyti praktinėje UAB „Exigen Services Lietuva“ veikloje.

Aprobacija. Darbas buvo aptartas įmonės UAB „Exigen Services Lietuva“ seminaruose ir pasitarimuose.

1. *Scrum* metodologijos analizė, privalumai ir trūkumai

1.1 Įvadas

Programinės įrangos kūrimo procesą sudaro tokios sudedamosios dalys kaip reikalavimai, kuriuos neteisingai supratus, programinė įranga yra kuriama netiksliai, jei klaidos pastebimos testavimo procese, ne visada spėjama jas išspręsti ir užbaigti programinės įrangos kūrimo procesą laiku. Stengiantis sumažinti galimų klaidų bei problemų tikimybę ir užtikrinti aukštą produkto kokybę yra naudojamos programinės įrangos kūrimo metodologijos, ir viena iš jų – *Scrum*.

Šio skyriaus tikslas – išanalizuoti *Scrum* metodologiją, kaip vyksta visas procesas naudojant būtent šią metodologiją, bei nustatyti šios metodologijos privalumus ir trūkumus.

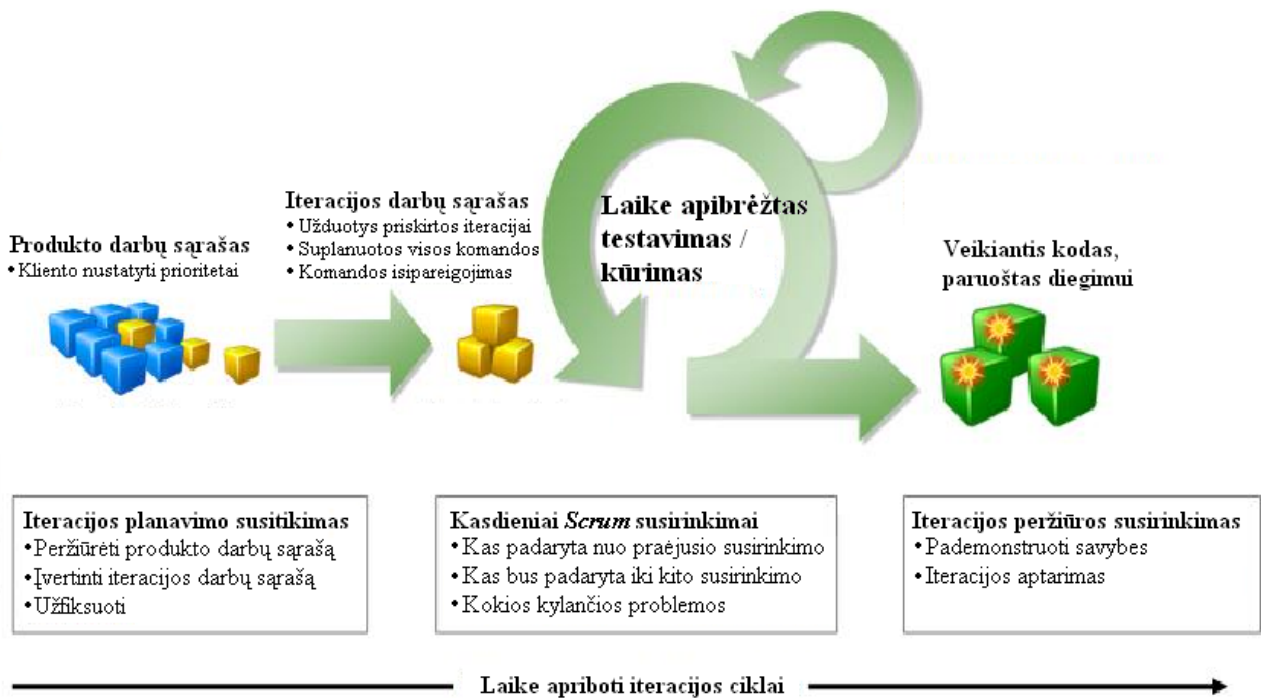
1.2 *Scrum* metodologijos analizė

Scrum – tai lanksčiojo programavimo projektų valdymo metodas, kai kuriais atvejais ženkliai pagerinantis komandų produktyvumą. Šis metodas yra skirtas kompiuterių programų, įvairių projektų ar produktų kūrimo valdymui.

Pagrindinės penkios charakteristikos, kuriomis, naudojant *Scrum*, lengviau užbaigti darbų sąrašą (angl. *living backlog*) susikaupusius darbus ir įvertinti jų prioritetus [2]:

1. Kiekvieną dieną, dažniausiai ryte, nustatytu laiku organizuojami *Scrum* susirinkimai, kuriuose kiekvienas komandos narys turi pasakyti, ką jis nuveikė per vakar dieną, ką planuoja atlikti šią dieną ir pasisakyti apie kliūtis, su kuriomis susiduria dirbdamas.
2. Kiekvienos iteracijos (angl. *sprint*) pradžioje arba pabaigoje yra rengiami iteracijos planavimo (angl. *sprint planning*) susirinkimai, kuriuose visi komandos nariai planuoja darbus naujai iteracijai.
3. Kiekvienos iteracijos paskutinę dieną yra rengiamas susirinkimas, kuriame yra apžvelgiami atlikti darbai, pasisakoma, kas praėjo gerai, kas praėjo blogai.
4. Daug laiko užimantys darbai turi būti padaryti per vieną ar dvi iteracijas.
5. *Scrum* vedlys (angl. *scrum master*) – žmogus, kuris turi stengtis, kad komanda spėtų įvykdyti visas iteracijai suplanuotas užduotis, kad užduotys būtų aiškios, ir padėti spręsti visas iškilusias problemas.

Scrum metodo proceso schema pateikiama 1pav.



1 pav. Scrum metodo procesas [9]

Pirmas etapas – yra sudaromas produkto darbų sąrašas, kuris susideda iš kliento reikalavimų. Kiekvienam reikalavimui yra suteikiamas prioritetą, kurį taip pat nustato klientas. Pagal prioritetus reikalavimai iš darbų sąrašo yra planuojami naujai iteracijai, preliminariai įvertinama, kiek užtruks suprogramuoti, ištestuoti ir įdiegti kiekvieną reikalavimą nuo pradžios iki galo, tik tada yra fiksuojamas sąrašas reikalavimų, kuris gali būti padarytas per vieną iteraciją. Suplanuotus reikalavimus komanda įsipareigoja užbaigti per vieną iteraciją. Iteracijos metu, kuri dažniausiai trunka nuo 2 iki 4 savaičių, vyksta kasdieniai *Scrum* susirinkimai, programuojami bei testuojami suplanuoti reikalavimai. Iteracijai baigiantis visi reikalavimai, kurie buvo suplanuoti, privalo būti galutinai užbaigti ir įdiegti arba paruošti diegimui, tai priklauso nuo situacijos ar susitarimo. Iteracijai pasibaigus, klientams ir kitiems asmenims, kuriems tai yra aktualu, demonstruojamos naujos sistemos savybės. Ir paskutinis etapas – iteracijos aptarimas arba kitaip retrospektyvos susirinkimas (angl. *retrospective meeting*). Šiame susirinkime aptariama, kas per iteraciją pasisekė gerai, kas buvo blogai ir galėtų būti pagerinta ateityje, kitose iteracijose [3].

Tiek kitur, tiek naudojant *Scrum* metodą, kiekvienai komandai, jos vadovui ir kitiems vadovams rūpi darbo pažanga, kaip sekasi viską įgyvendinti, taigi yra išskiriami trys labiausiai dominantys klausimai, susiję su darbo progresu [4]:

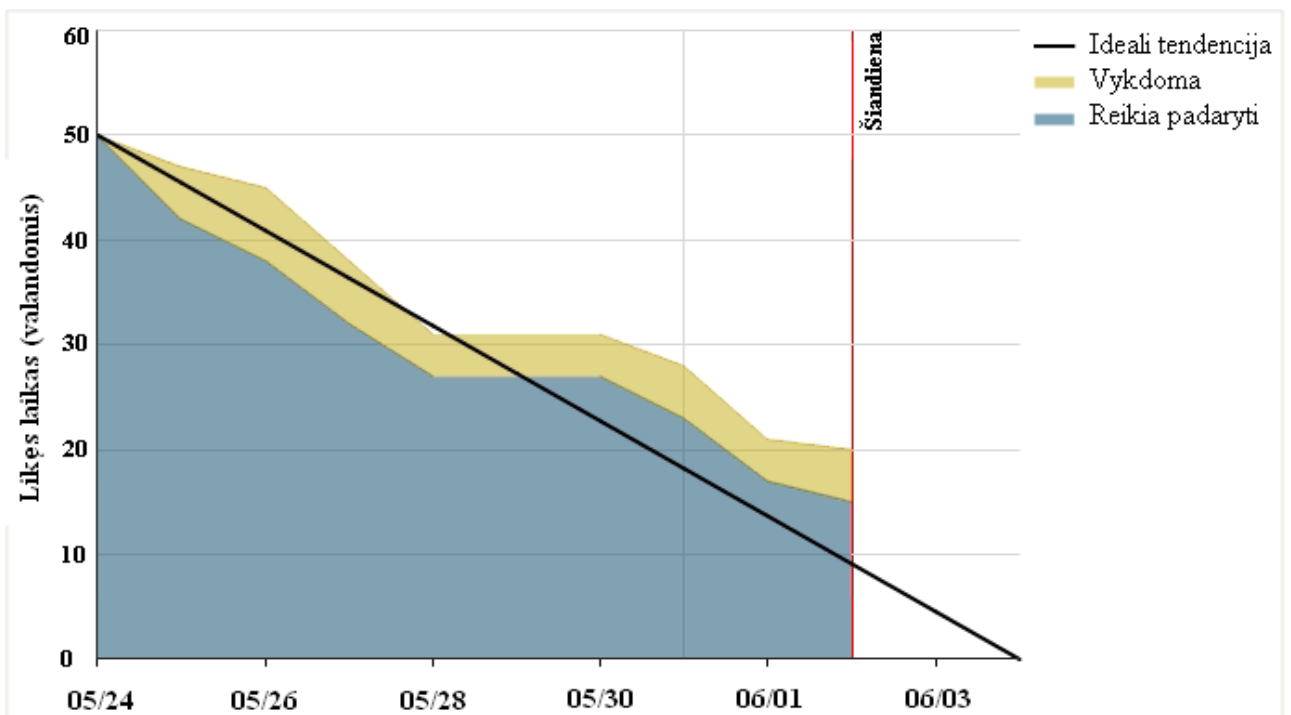
- 1) iteracijos progresas – kiek darbų jau galutinai užbaigta, kiek vis dar reikia padaryti ir kiek vis dar yra daroma;
- 2) produkto progresas – kaip sekasi kurti galutinį produktą, ar neatsilieka nuo numatytos užbaigimo datos;
- 3) naujos versijos išleidimo progresas – ar galutinis produkto variantas su teisingai ir kokybiškai įgyvendintais reikalavimais klientą pasieks laiku.

Į visus šiuos klausimus atsakyti yra pateikiamos aiškos diagramos, iteracijos progreso diagrama (angl. *sprint burndown chart*), produkto progreso diagrama (angl. *product burndown chart*) bei naujos versijos išleidimo diagrama (angl. *release burndown chart*), kurios atvaizduoja visus progressus.

2 pav. pavaizduota iteracijos progreso diagrama parodo, kiek laiko lieka nurodytų intervalų pabaigoje per iteraciją. Horizontalioji ašis rodo iteracijos dienas, o vertikalioji – darbo kiekį valandomis, kuris lieka, kad visos užduotys būtų sėkmingai užbaigtos per iteraciją.

Iteracijos progreso diagrama atvaizduoja šiuos duomenis [10]:

- Ideali tendencija – ši linija rodo idealią situaciją, jei visas darbas vyktų be jokių netikėtumų ar trukdžių. Komanda deda visas pastangas, kad bent jau iteracijos pabaigoje padaryti darbai sutaptų su idealia linija.
- Vykdomo dalis parodo, kiek valandų lieka užduotims atlikti, kurios yra jau pradėtos ir vykdomos prasidėjus iteracijai.
- Baigimo dalis parodo, kiek valandų lieka užduotims atlikti, kurios dar nėra pradėtos, bet turi būti užbaigtos iteracijai pasibaigus.



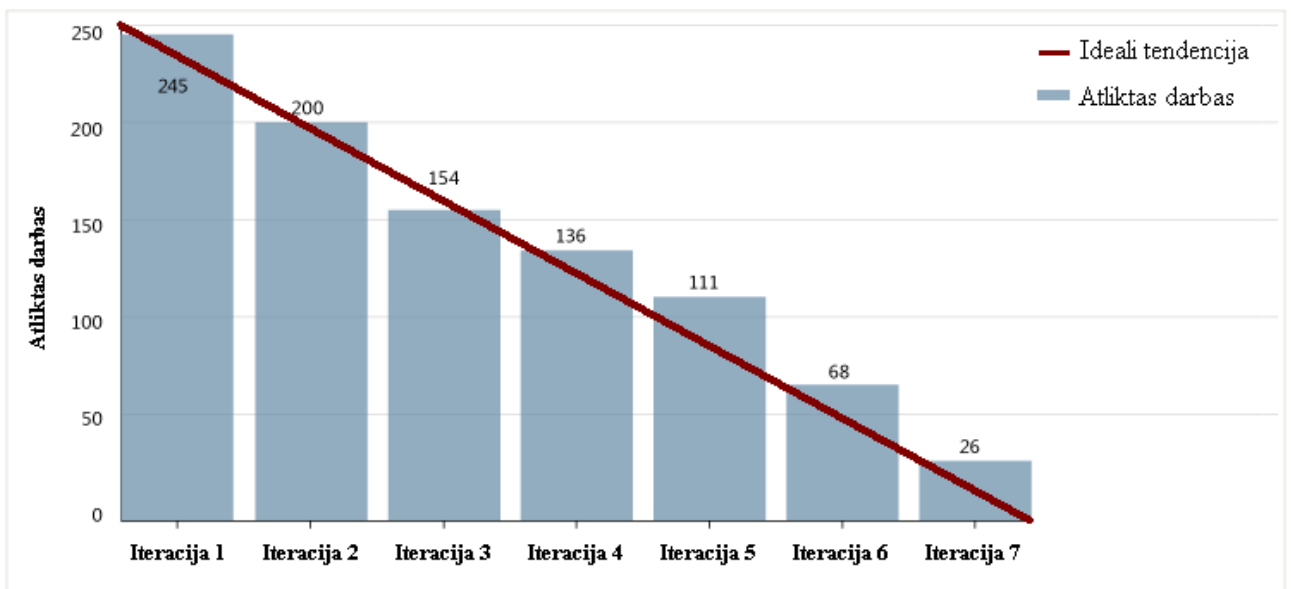
2 pav. Iteracijos progreso diagrama [10]

3 pav. pavaizduota naujos versijos išleidimo diagrama taip pat parodo, kiek lieka laiko atlikti darbams, kad nauja versija galėtų būti išleista laiku. Horizontalioje ašyje rodomos iteracijos, o vertikalioje atliktas darbas per vieną iteraciją.

Naujos versijos išleidimo diagrama atvaizduoja šiuos duomenis [11]:

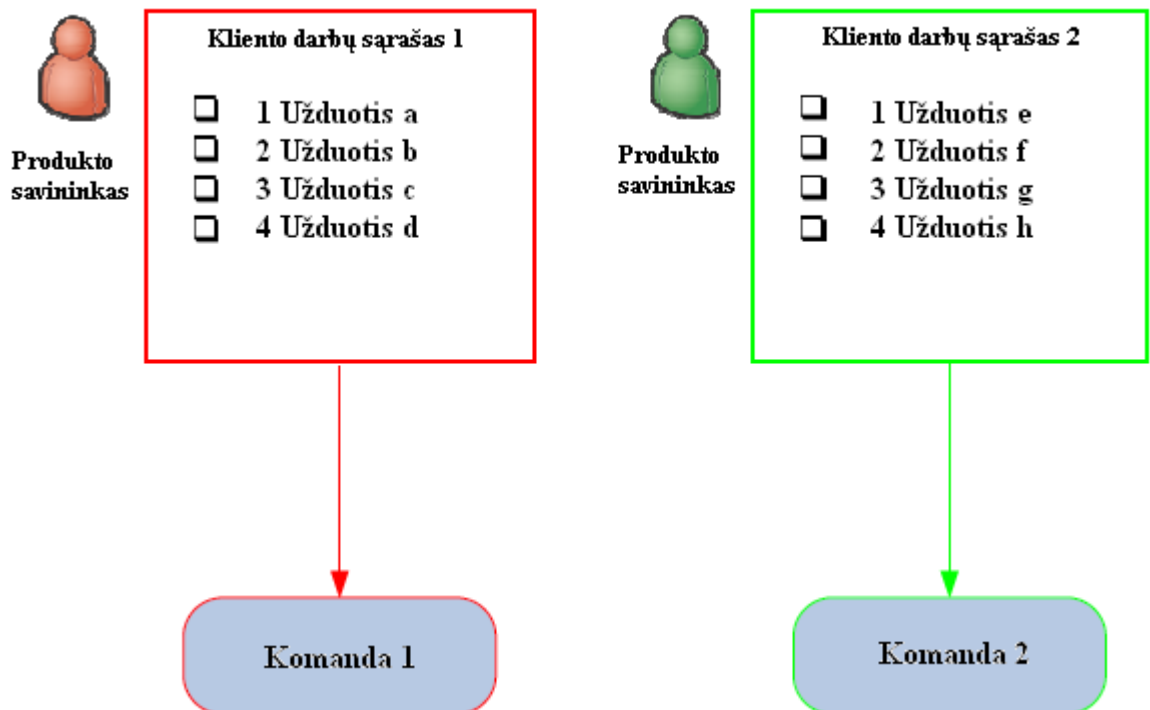
- Ideali tendencija – ši linija, kaip iteracijos progreso diagramoje, rodo idealią situaciją, jei visas darbas vyktų be jokių netikėtumų ar trukdžių. Komanda deda visas pastangas, kad kiekvienos iteracijos pabaigoje atlikti darbai sutaptų su idealia linija.
- Atliktas darbas – tai darbo kiekis, kuris yra atliekamas per vieną iteraciją. Jis gali būti vertinamas valandomis arba istoriniais taškais (angl. *story points*).

Produkto progreso diagrama iš esmės sutampa su naujos versijos išleidimo diagrama, esminis skirtumas yra tik tas, kad diagrama atvaizduoja, kaip sekasi siekti sėkmingo produkto, o ne naujos versijos, užbaigimo.



3 pav. Naudos versijos išleidimo diagrama [11]

Kuriant programinę įrangą, vieni projektai yra mažesni, kiti – didesni. Sėkmingai įgyvendinti mažesnius projektus pakanka vienos savarankiškos komandos, sudarytos iš nedidelio žmonių skaičiaus, o dideliems reikia kur kas daugiau žmonių, todėl kartais nėra patogu, jog visi prie projekto dirbantys žmonės priklausytų vienai komandai. Tokiu atveju komandos yra skirstomos į keletą mažesnių komandų, o suskirstymo būdai priklauso nuo projekto dydžio.



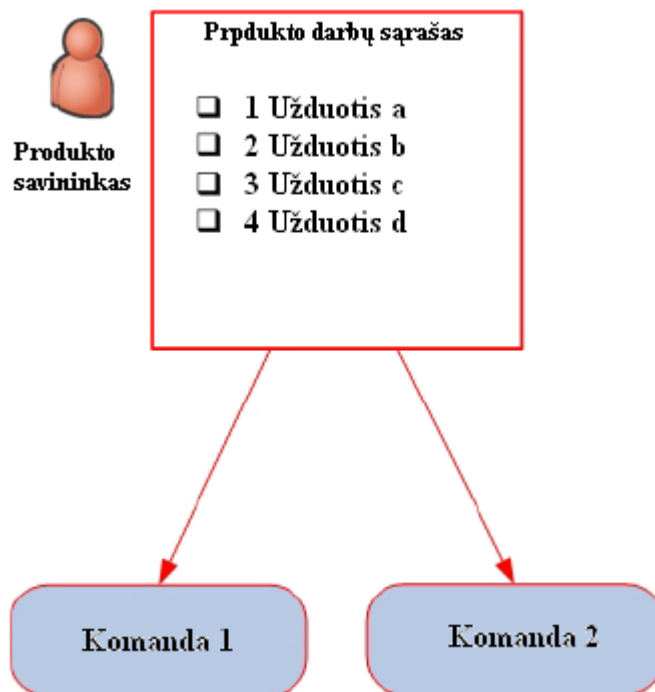
4 pav. Nepriklausomos komandos [17]

4 pav. pavaizduotos nepriklausomos komandos. Joms dažniausiai skiriami maži projektai. Jos pasižymi tokiomis ypatybėmis [17]:

- Esant skirtingiems klientams, reikalavimai planuojami ir prioritetai parenkami nepriklausomai.
- Atskirai planuojami darbai iteracijoms.
- Atskiri produkto darbų sąrašai.
- Atskiri iteracijų darbų sąrašai.

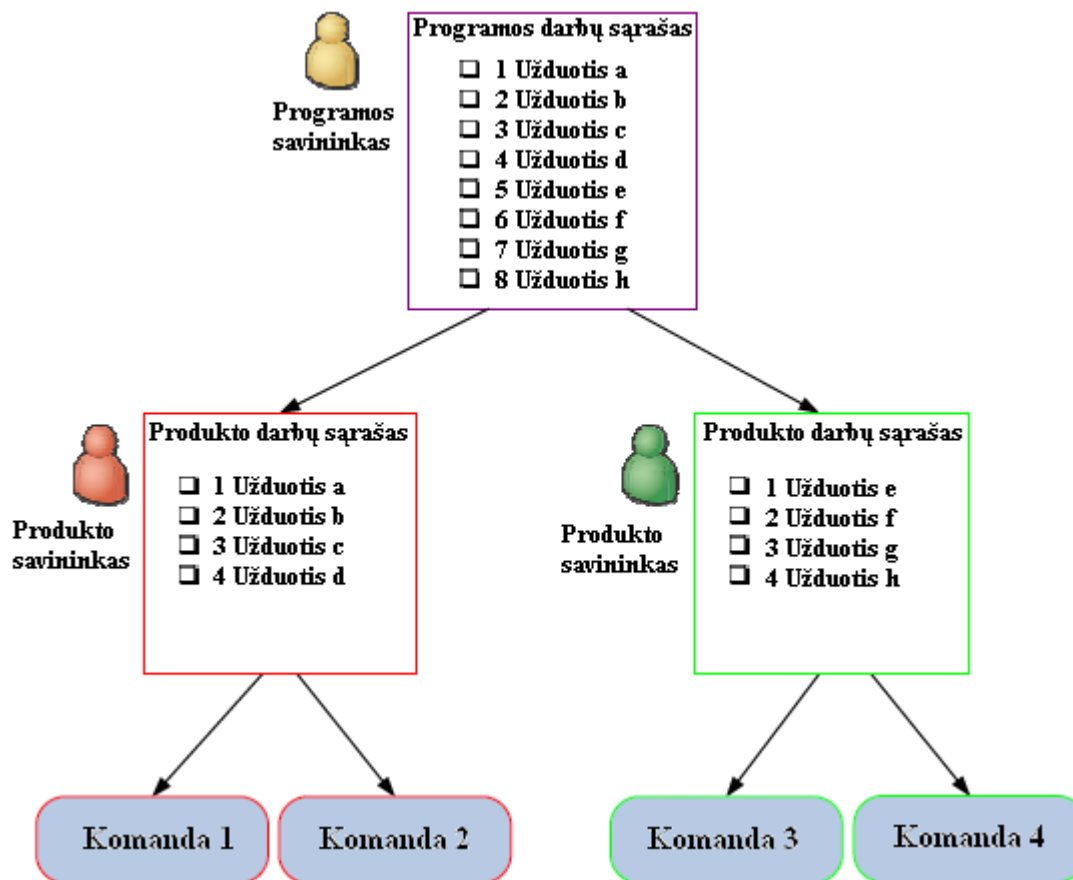
Didesniuose projektuose, kuriuose dalyvauja nuo 1 iki 3 komandų, yra naudojamas 1-ojo lygio skalinis modelis, kuris atvaizduojamas 5 pav. Šiam modeliui būdingos šios savybės [17]:

- Vienas klientas, kuriam dirba visos komandos.
- Vienas prioritetų parinkimo reikalavimų rinkinys.
- Bendras visų komandų iteracijų planavimas ir peržiūros.
- Atskiri iteracijų darbų sąrašai.
- Dinaminis komandų suskirstymas nuo iteracijos iki iteracijos.
- Modelis sklandžiai veikia, esant nedideliame komandų skaičiui.
- Kiekvienoje komandoje yra iteracijos planai ir peržiūros, kasdieniai *Scrum* susirinkimai, bendras programos kodas ir jo tvirtinimas.



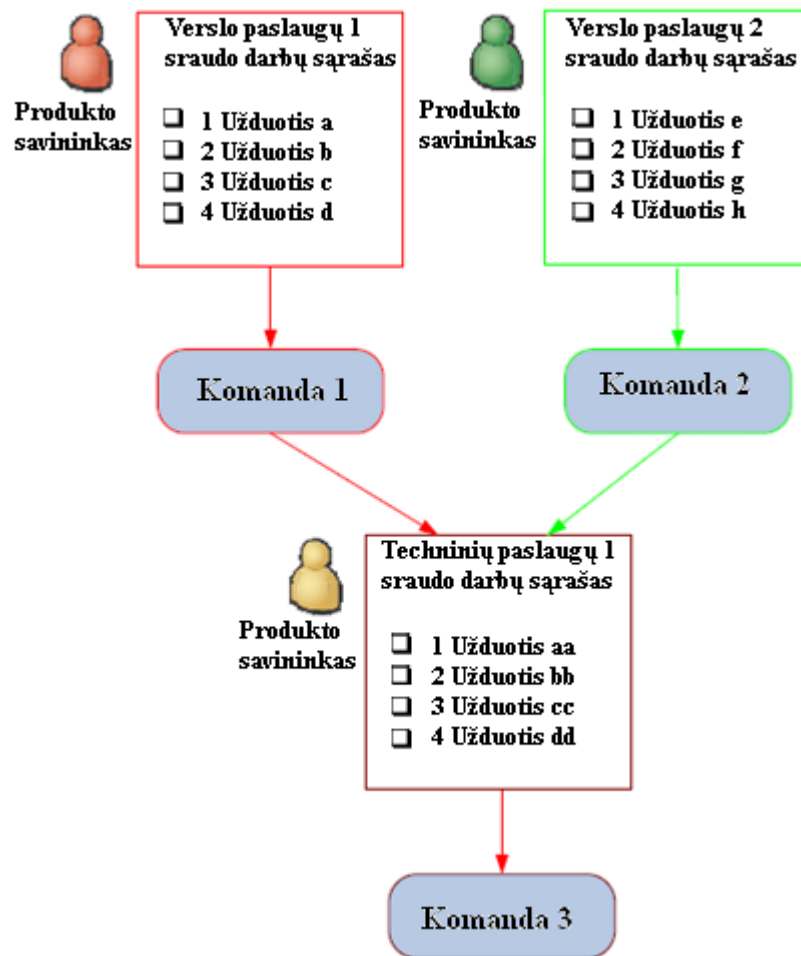
5 pav. 1-ojo lygio skalinis modelis [17]

Projektuose, kuriuose komandų skaičius viršija 3, yra naudojamas 2-ojo lygio skalinis modelis, pavaizduotas 6 pav. Šis modelis yra tarsi keli 1-ojo lygio skaliniai modeliai, kadangi programos darbų sąrašas yra labai didelis. Jis suskirstomas į vienodas dalis, tarkim, pagal verslo paslaugas (pvz., CRM) ir technines paslaugas (pvz., tinklo platformą), ir tos dalys ir padalinamos vienai komandų grupei ir kitai komandų grupei, tokiu būdu yra sukuriami du 1-ojo lygio skaliniai modeliai, dirbantys, vienam tikslui [17].



6 pav. 2-ojo lygio skalinis modelis [17]

Įgyvendinant įvairius projektus, kartais nutinka taip, jog vienos komandos būna priklausomos nuo kitų komandų, taigi komandos tarsi tampa klientais tų komandų, nuo kurių yra vienaip ar kitaip priklausomos (7 pav.). Taip gali nutikti, kai kelios komandos dirba prie vieno projekto, tarkim, pagal 2-ojo lygio skalinį modelį, ir vienai komandai prireikia įgyvendinti užduotį iš savo darbų sąrašo, kuri neveiks arba veiks neteisingai, jei nebus įgyvendinta užduotis, esanti kitos komandos darbų sąrašė. Tokiu atveju komandai, priklausomai nuo kitos komandos, užduoties įgyvendinti negalima, reikia laukti, kol kita komanda įgyvendins savo užduotį arba abiem komandoms reikia įgyvendinti savo užduotis kartu [17].



7 pav. Komandos kaip „Klientai“ [17]

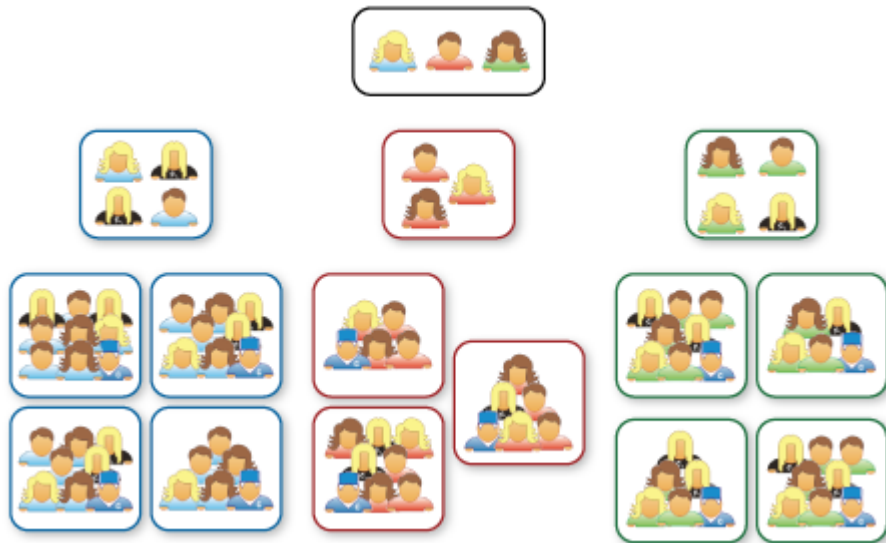
Modelio savybės [17]:

- 1-a ir 2-a komandos yra vadinamieji 3-os komandos klientai.
- 3-a komanda turi produkto savininką, kuris skirsto techninio produktų sąrašo prioritetus.
- 3-a komanda konsultuojasi su 1-a ir 2-a komandomis, kol diegiamos kitos užduotys.
- 3-a komanda peržiūri savo iteracijos planą.
- 1-a ir 2-a komandos gali pateikti savo pastabas.
- 1-a ir 2-a komandos integruoja 3-os komandos padarytus atnaujinimus tam tikromis iteracijomis.
- 3-a komanda gali dirbti mažesnėmis iteracijomis, bet vis dar kartu su 1-a ir 2-a komandomis.

- 3-a komanda gali pasiskolinti 1-os ir 2-os komandų narius, susitarusi per iteracijos planavimą.

Esant dideliame komandų skaičiui, bendravimas tarpusavyje yra būtinas, todėl tam puikiai tinka *scrum of scrums* susirinkimai. Šie susirinkimai leidžia komandų grupėms aptarti jų vykdomus darbus, ypač atkreipiant dėmesį į sutampančias ir integracijos vietas. Tarkime, yra projektas, kurį sudaro tam tikras skaičius komandų, o kiekviena komanda yra sudaryta iš tam tikro skaičiaus žmonių. Kiekviena komanda rengia (vieni metu arba paeiliui) savo kasdienes *scrum* susirinkimus. Tuomet kiekviena komanda paskiria žmogų, kuris turės dalyvauti *scrum of scrums* susirinkime. Sprendimą, kas dalyvaus *scrum of scrums* susirinkime, kiekviena komanda priima individualiai. Paprastai pasirinktas asmuo būna tam tikros srities žinovas – programuotojas, testuotojas, duomenų bazių administratorius ir panašiai, tai neturėtų būti produkto savininkas ar *scrum* vedlys. Žmogus, kuris yra išrenkamas dalyvauti *scrum of scrums* susirinkime, nebūtinai turės jame dalyvauti nuolatos. Dalyviai karts nuo karto turi keistis. Žmonėms besikeičiant, komanda turi nuspręsti, kuris jų visų poziciją atstovaus geriausiai, kuris geriausiai supras ir galės pakomentuoti esamas problemas, kad komanda galėtų sėkmingai dirbti. Vieną dieną komandoje gali būti techninių problemų, todėl reikėtų siųsti žmogų, kuris labiau nusimanytų apie techninius dalykus, kitą dieną gali būti daugiau problemų, susijusių su testavimu, todėl reikėtų siųsti žmogų, išmanantį testavimą. Jei projekte esančių komandų skaičius yra nedidelis (rekomenduojama ne daugiau kaip keturios komandos), tuomet į *scrum of scrums* susirinkimus gali eiti du žmonės, tarkim, išmanantis techninę sritį ir *scrum* vedlys, jei taip nusprendžia komanda [4].

Scrum of scrums susirinkimai gali būti suskirstomi rekursiniu būdu. Tarkime, organizacijoje kiekvieną dieną vyksta septyni atskiri *scrum of scrums* susirinkimai. Kiekviename iš septynių *scrum of scrums* dalyvauja po vieną asmenį iš skirtingų komandų. Toks *scrum of scrums* susirinkimų darbas gali būti koordinuojamas dar aukštesniu lygiu: *scrum of scrums of scrums*. Kaip visa tai turėtų atrodyti yra pateikiama 8 pav [12].



8 pav. *Scrum* susirinkimų lygmenys [12]

Anot Keno Schwaberio (*Scrum* metodologijos pradininko), *scrum of scrums* susirinkimai turėtų vykti kasdien, kaip ir kasdieniai *scrum* susirinkimai, bet tai tik rekomendacija, kadangi susirinkimų dažnumą privalo nuspręsti pačios komandos. Taip pat yra siūloma, kad tiek *scrum*, tiek *scrum of scrums* susirinkimai neužsitęstų ilgiau nei 15 min., kadangi tiek laiko visiškai užtenka aptarti esminiams klausimams, į kuriuos per kiekvieną susirinkimą privalo atsakyti kiekvienas susirinkimo dalyvis, o esmines problemas, apie kurias yra pasakoma per susirinkimą, reikia aptarti po susirinkimo, ir tik tiems žmonėms, kurie su jomis susiję, kad kiti galėtų eiti toliau dirbti ir negaišti laiko visiškai be reikalo [4].

Klausimai pateikiami kiekvienam dalyvaujančiajam *Scrum* susirinkime:

- Ką nuveikė per vakar dieną?
- Ką žadi nuveikti per šią dieną?
- Ar susiduri su kokiomis nors problemomis, kurios trukdytų dirbti?

Klausimai pateikiami kiekvienam dalyvaujančiajam *Scrum of scrums* susirinkime:

- Ką tavo komanda nuveikė nuo paskutinio susirinkimo?
- Ką tavo komanda planuoja nuveikti iki mūsų sekančio susirinkimo?
- Ar yra kokių problemų, kurios trukdytų ar lėtintų tavo komandos darbą?
- Ar ketinate ką nors implementuoti, kas būtų aktualu kitoms komandoms?

1.3 Scrum metodologijos trūkumai

Ko gero kiekviena egzistuojanti metodologija nėra tobula, kaskart naudojant vieną ar kitą metodologiją, laikui bėgant atsiranda vienokių ar kitokių trūkumų, kuriuos reikia kažkaip spręsti. Ne išimtis ir *Scrum* metodologija, kuri taip pat pasižymi tam tikrais trūkumais įvairiose situacijose [13], [14]:

- *Scrum* metodas yra viena iš pagrindinių priežasčių, kodėl kartais padaromų darbų apimtis juda gan lėtai. Taip yra todėl, kad naudojant *scrum* metodą, darbai suskirstomi iteracijomis, kur kiekviena iš jų turi suplanuotą tam tikrą kiekį darbų (ne daugiau ir ne mažiau) iki tam tikro laiko. Tuo tarpu užbaigus visus darbus iteracijai dar nespėjus pasibaigti, nauji darbai negali būti pradėti, nes tai jau nebėra šios iteracijos apimtyje, o suinteresuotos šalys nepaliaujamai reikalauja, kad komandos įgyvendintų ir naujus funkcionalumus.
- Jei užduotis, kuri reikia atlikti, nėra aiškiai pateikta, tuomet jos planavimas užtrunka ilgiau bei laikas negali būti įvertintas tiksliai, o tai kainuoja papildomus pinigus. Tokiu atveju užduotį gali tekti įgyvendinti per kelias iteracijas.
- Jei komandos nariai, ar bent vienas narys, nespėja užbaigti savo užduoties iki galo iki iteracijos pabaigos, tuomet iteracija skaitoma nepavykusia ir neužbaigta. To pasekoje, klientas greičiausiai liks nepatenkintas, o komandai gali būti taikomos vienokios ar kitokios sankcijos.
- Jei bent vienas komandos narys netikėtai, dėl pateisinamų priežasčių, pasitraukia iš komandos nors keletai dienų per programavimo procesą, tai gali turėti didelių pasekmių sėkmingam iteracijos užbaigimui, kadangi darbai būna suplanuoti kiekvienam darbuotojui nuo iteracijos pradžios iki pabaigos.
- Naudojant šį metodą patartina, kad visi, ar bent jau didžioji dauguma komandos narių, būtų turintys patirties savo srityje, kadangi priešingu atveju, jei komandos bus sudarytos iš patirties neturinčių, ar mažą patirtį turinčių žmonių, kyla grėsmė, jog projektai nebus sėkmingi, darbai nebus spėjami padaryti per iteracijas ir t.t.

1.4 Scrum metodologijos privalumai

Apžvelgus bene labiausiai pastebimus *scrum* metodologijos trūkumus reiktų paminėti ir metodologijos privalumus, juk ši metodologija nebūtų tokia populiari, ir apskritai turbūt nebūtų naudojama, jei jos trūkumai būtų didesni už privalumus. Taigi, kaip ir daugelis gerai žinomų

metodologijų, taip ir *scrum* metodologija turi ją išskiriančius privalumus, kurie ir skatina žmones ją naudoti [15], [16]:

- Tinkamai naudojant *scrum* metodą pirmiausia klientui bus pristatomi didžiausios svarbos ir didžiausią prioritetą turintys programos atnaujinimai, taip išvengiant laiko gaišimo programuojant tuos atnaujinimus, kurie, galbūt, niekada nebus naudojami.
- Daugumoje kompanijų programavimas yra pristabdomas dėl problemų, kurios vadinamos kliūtimis (angl. *impediments*). Naudojant *scrum* šios kliūtys yra prioritizuojamos ir sistemingai pašalinamos, atsižvelgiant į svarbą ir toliau didinant produktyvumą bei kokybę.
- Pašalinamas vadovybės spaudimas komandoms. Kiekviena komanda gali skirstytis ir planuoti darbus taip, kaip joms atrodo geriausia, pagal esamą situaciją.
- *Scrum* susirinkimai yra didžiulis pliusas, kurio dėka komandos viduje ir tarp komandų vyksta daugiau bendravimo. Komandos viduje kiekvieną dieną vykstančių susirinkimų dėka, visi komandos nariai nuolatos žino, ką tą dieną veikia kiti komandos nariai, nesidubliuoja darbai, garsiai kalbama apie problemas, kurios trukdo darbui ir t.t.
- Paprastos *scrum* metodo taisyklės leidžia nuolatinį tikrinimą, pritaikymą, saviorganizaciją bei naujovių atsiradimą, ko dėka produktas klientui gali būti pateikiamas tik geresnis ir aukštesnės kokybės.
- Darbo našumas ir pasitenkinimas darbu – abu punktai reikalauja, kad žmonės dirbtų produktyviai, taigi dažniausiai yra dirbama pagal dvi strategijas:
 - Paralelinis darbas su viena užduotimi, kai prie vienos užduoties dirba visa komanda ir kol užduotis neužbaigiama, tol nepradedama nauja.
 - Paralelinis darbas su skirtingomis užduotimis, kai prie skirtingų užduočių dirba visa komanda.

1.5 Išvados

Išnagrinėjus *scrum* metodologiją, jos trūkumus bei privalumus, nustatyta, kad ši metodologija yra sukurta taip, kad komandų darbas būtų kaip įmanoma labiau optimizuotas, galutinio produkto kokybė kuo aukštesnė, siekiant sumažinti išlaidas ir gauti didžiausią naudą. Metodologija yra puikiai tinkanti įvairiems projektams įgyvendinti, tačiau turi keletą trūkumų, kurie ne visada, bet kartais gali trukdyti sėkmingam ir kokybiškam komandos darbui.

2. *Kanban* metodologijos analizė, privalumai ir trūkumai

2.1 Įvadas

Kiekvienos informacinių technologijų įmonės vienas iš pagrindinių tikslų yra užtikrinti kuo didesnę darbuotojų našumą ir kuo aukštesnę parduodamo produkto kokybę. Siekti šių tikslų padeda įvairios darbo metodologijos, tačiau laikui bėgant keičiasi daugybė veiksnių, kurie vėliau skatina ieškoti alternatyvų senai naudojamoms metodologijoms. Viena iš alternatyvų kitoms metodologijoms galėtų būti *Kanban* metodologija, kuri skatina visą komandą dirbti kartu.

Šio skyriaus tikslas – išanalizuoti *Kanban* metodologiją, privalumus bei trūkumus.

2.2 *Kanban* metodologijos analizė

Į daugumą projektų galima žiūrėti kaip į procesus, kurie vyksta įgyvendinant eiles žingsnių ar užduočių, kuriuos įvykdžius gaunamas norimas rezultatas. Yra įvairiausių procesų – paprastų ir sudėtingų, individualių ir komandinių, greitų ir reikalaujančių nemažai laiko. Kartais dideli ar visaapimantys procesai susideda iš keleto mažesnių procesų.

Kanban – tai metodologija, skirta projektų, kurie dažniausiai kuria įvairias sistemas naudodamiesi lanksčiojo programavimo metodu, informacijos, duomenų ar veiklos procesų valdymui. *Kanban* metodologija remiasi *Lean* metodologijos principais ir ganėtinai dažnai yra laikoma supaprastinta *Scrum* metodologijos versija [20].

Nepakankamas duomenų kiekis, nesvarbu ar tai yra kažkokia funkcionalumo dalis, ar dokumentas, ar kliento informacija, tuo metu, kai tų duomenų prireikia, o jų tiesiog nėra, tai sukelia bereikalingą laiko švaistymą ar vėlavimą, atiduodant programinę įrangą klientui. Kita vertus, turint per daug pradėtų ir neužbaigtų darbų irgi yra tam tikra laiko švaistymo forma. *Kanban* – tai metodologija, leisianti išmokti, kaip optimaliai suvaldyti darbo tėkmę proceso metu. Trys pagrindinės charakteristikos, kuriomis yra pagrįsta *Kanban* metodologija [7]:

- **Darbo eigos išryškėjimas**
 - Darbo suskirstymas į smulkesnes dalis, kiekvienas punktas yra užrašomas ant kortelės, kuri yra priklijuojama prie sienos ar lentos.
 - Siena ar lenta yra suskirstyta į stulpelius su pavadinimais, kad būtų aiškiai suprantama, kas yra atliekama su kiekvienu punktu darbo eigos metu.

- **Neužbaigtų darbų apribojimas** (angl. *work in progress limits, WIP limits*) – kiekvienam stulpeliui yra priskiriamas tikslus apribojimas, kiek punktų gali būti pradėtų ir neužbaigtų darbo eigos metu.
- **Proceso tėkmės vertinimas ir pagerinimas** (vidutinis laiko tarpas, skirtas užbaigti vienam punktui, kartais vadinamas “ciklo laiku”), optimizuoti procesą taip, kad pagrindinis laikas, skirtas punktų įgyvendinimui, būtų kaip įmanoma trumpesnis ir kuo labiau nuspėjamas.

Darbo eigos išryškinimas – tai vaizdinis proceso atvaizdavimas lentoje (9 pav.), kuris leidžia stebėti, kaip tiksliai užduotys ar punktai keičiasi nuo “Reikia atlikti” statuso iki “Atlikta”.

Kuo procesas sudėtingesnis, tuo vaizdinio proceso atvaizdavimo kūrimo eiga tampa naudingesnė ir svarbesnė. *Kanban* lenta nebūtinai turi būti sudaryta iš tokių stulpelių kaip pateikta 9 pav. *Kanban* metodologija kuo puikiausiai gali būti naudojama ir tuomet, jei yra vos keletas žingsnių (stulpelių) kuriant programinę įrangą (reikia atlikti, atliekama, atlikta) arba daug žingsnių (planavimas, projektavimas, rengimas, patvirtinimas, planavimas, įgyvendinimas, testavimas, integravimas, diegimas). Taigi, kad ir koks sudėtingas projektas bebūtų, turima *Kanban* lenta leidžia pamatyti atliekamų ir turimų atlikti darbų statusus akimirksniu [1].



9 pav. *Kanban* lenta (angl. *Kanban board*) [1]

Kanban lenta – tai, ko gero, pirmas dalykas, kurį reiktų įgyvendinti, norint pradėti naudoti *Kanban* metodologiją. Norint sudaryti tokią lentą atitinkamam projektui, pirmiausia reiktų

nustatyti pagrindinius procesus (10 pav.), kurie vyksta visoje organizacijoje ar projekte, o tuomet nustatyti atitinkamus žingsnius kiekvienam procesui individualiai. Detalizuojant kiekvieną procesą, reikėtų pagalvoti apie tokius punktus, kaip [1]:

- Kas sudarinėja reikalavimus?
- Kas nustato reikalavimų prioritetus, apibrėžtumą ir paskirstymą?
- Kokie yra pagrindiniai žingsniai, kurie lemia sėkmingą užduoties įgyvendinimą?
- Kiek žmonių bus tiesiogiai prisidedančių prie sėkmingo užduoties įgyvendinimo?

Visi šie ir daugybė panašių punktų yra pradinė eiga, sudarinėjant *Kanban* lentą. Kiekvienas sugalvotas žingsnis gauna vietą *Kanban* lentoje.

Keletas tradicinių *Kanban* lentos pavyzdžių, kurie naudojami ir kasdieniniame gyvenime, ir programinę įrangą kuriančiose įmonėse [1]:

- *Namų ruošą*: Reikia padaryti – Atliekama – Atlikta
- *Pardavimų rinka*: Vadovo parengimas – Vadovo pateikimas – Pardavimų pristatymas – Pasiūlymas – Derybos – Sutarties pasirašymas/ nutraukimas – Tolesni veiksniai – Palaikymas ir techninė priežiūra
- *Programinės įrangos kūrimo projektai*: Darbų, kuriuos reikia atlikti, sąrašas – Reikalavimai – Projektavimas – Programavimas – Testavimas – Atlikto darbo įvertinimas ir priėmimas – Diegimas – Palaikymas

Sudarinėjant procesų žingsnius yra labai svarbu juos sudarinėti pagal esamą procesą, o ne pagal kažkokį idealų proceso pavyzdį ar pagal tokį, kokio norėtusi. Pradinis sudarinėjimo taškas turi būti procesas, kuris iš tikrųjų egzistuoja realiame pasaulyje. Jei nutiktų taip, kad proceso eigoje sumažėtų efektyvumas, susidarytų sangrūdos (angl. *Bottlenecks*) arba atsirastų trūkstamų ar nereikalingų žingsnių, visa tai anksčiau ar vėliau iškils į dienos šviesą, tuomet proceso eiga galės būti pakeista ir pritaikyta taip, kaip geriausia visai komandai [7].

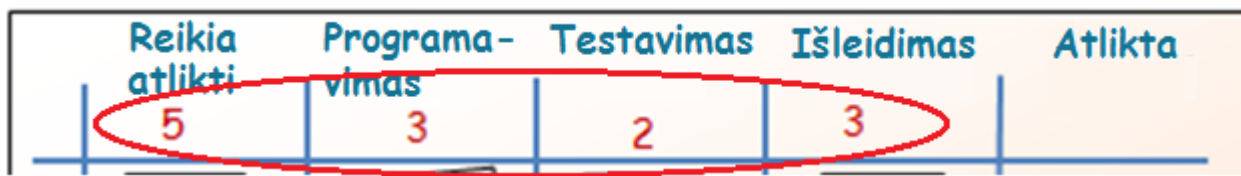


10 pav. Pagrindiniai darbo procesai [7]

Prieš sudarant kiekvieną žingsnį reikia jį gerai apsvarstyti ir nustatyti, kiek užduočių, tame žingsnyje (lentos stulpelyje), bus galima atlikti vienu metu. Pavyzdžiui, vienas programuotojas, vienu metu, ko gero, gali susitelkti ir susikoncentruoti ties viena nauja užduotimi (angl. *New*

feature), vienas testuotojas, galbūt įstengtų vienu metu testuoti keletą naujų užduočių, na, o žmogus, dirbantis su pardavimais, greičiausiai įstengtų susitvarkyti ir su trim ar net keturiomis perspektyvomis vienu metu. Visa tai vadinama neužbaigtų / pradėtų darbų apribojimu.

Neužbaigtų darbų apribojimas – tai *Kanban* metodologijos dalis, kurios dėka yra atliekama daugiau darbų, nors pastangų įdedama mažiau. Visa tai gali skambėti kiek prieštaringai ir tarsi tai būtų neįmanoma, bet tai iš tiesų yra galinga idėja, kuri laikui bėgant vis labiau ir labiau pasitvirtina. Kiekvienas *Kanban* lentos stulpelis yra pažymėtas skaičiumi (10 pav.), kuris nurodo, kiek daugiausia neužbaigtų užduočių vienu metu gali būti atliekama kiekviename stulpelyje. Šis skaičius nebūtinai turi būti tiksliai nustatomas iš pirmo karto, pradžioje galima tiesiog spėti, įvertinus visus žingsnius, tačiau reikėtų nepamiršti, kad pradinis skaičius neturėtų būti labai didelis, kadangi bus sunkiau nustatyti ir atskleisti visus trūkumus. Nesvarbu ar projektas yra paprastas ar sudėtingas arba ar komanda yra didelė, ar maža, kadangi kiekvienu atveju yra pritaikomas optimalus atliekamo darbo kiekis, kuris gali būti atliekamas vienu metu, neaukojant darbo efektyvumo. Vienu metu atlikinėti dešimtis lygiaverčių darbų nėra nieko blogo, tačiau tai, ko gero, užimtų savaitę, ar daugiau, brangaus laiko, tuo tarpu atlikinėti kelis darbus vienu metu užimtų keletą valandų ar pusdienį, dirbant tokiu principu savaitės gale būtų galima tikėtis tikrai geresnio rezultato, nei dešimt atliktų darbų [7].



11 pav. *Kanban* lenta [1]

Jei komanda ar projekte dirbantys darbuotojai į pokyčius žiūri palankiai, tuomet *Kanban* galima pradėti naudoti nustatant mažesnę neužbaigtų darbų apribojimo limitą, taip iškart sukuriant iššūkį tapti kuo efektyvesniais. Dirbant su mažesniais neužbaigtų darbų apribojimo limitais, “galvos skausmą sukeliančios vietos” (sagrūdės, atidėliojimai ir pan.) proceso metu bus atskleidžiamos greičiau, o tai suteiks galimybę greičiau viską pagerinti.

Vis dėlto, jei komanda į pokyčius žiūri prieštaringai, tuomet dauguma komandoje dirbančių žmonių greičiausiai ims ignoruoti neužbaigtų darbų apribojimo limitus, kurie yra per maži ir tokiu atveju nieko neišmoks, o nauda tikrai bus nedidelė. Tokiu atveju limitus reikėtų padidinti iki vidutinio arba kiek didesnius, nei vidutinio dydžio, taip komandos nariams bus lengviau prisitaikyti

stebėti visą eigą. Kai komanda apsipranta prie esamų limitų, tuomet po truputį galima limitus mažinti, taip nesukeliant papildomo streso ar pasipriešinimo į pokyčius nusistovėjusioje rutinoje.

Dar vienas svarbus aspektas, stengiantis įgyti palankų komandos narių požiūrį, yra stengtis įtraukti ir visus komandos narius apibrėžiant ir dokumentuojant procesą, taip pat, kaip ir nustatinėjant neužbaigtų darbų apribojimo limitus, prie kurių komanda būtų linkusi taikytis ir dirbti pagal juos kiekviename žingsnyje. Labai svarbu įsitikinti, kad komanda supranta, jog apibrėžiant maksimalų darbų kiekį, kurį jie galės atlikti vienu metu, turi du pagrindinius privalumus:

- Tai sutrumpina laiką, kurio reikia, kad būtų galima atlikti bet kurią užduotį
- Tai pagerina kokybę, kadangi dėmesys yra sutelkiamas į keletą užduočių

Kartu šie privalumai pagerina efektyvumą, taiga komanda kartu gali nuveikti kur kas daugiau darbų. Darbai atliekami ne tik greičiau, bet ir geriau, nei kada nors anksčiau, o tai reiškia, jog bus mažiau papildomo darbo.

Kai neužbaigtų darbų apribojimo limitas yra nustatytas kiekvienam proceso etapui, tuomet šie skaičiai yra surašomi ties kiekvienu stulpeliu *Kanban* lentoje. Šie skaičiai turi būti aiškiai matomi kiekvienam komandos nariui, o pastarieji privalo dėti visas pastangas, kad šie skaičiai nebūtų viršyti. Gali būti nustatomos taisyklės, kurios nusakys, ką daryti tuo atveju, jei kas nors nori nesilaikyti šio limito, tokiu atveju gali būti rengiami komandos susirinkimai, kuriuose tokie sprendimai turėtų būti aiškiai pagrindžiami.

Kai *Kanban* lenta yra sudaryta, stulpeliai suskirstyti į žingsnius, o neužbaigtų darbų apribojimo limitai nustatyti, tuomet belieka ant jos pradėti dėti užduotis. Užduotys turėtų atspindėti tai, kas turi būti padaryta ir tai, ką privaloma padaryti, taip pat kiekviena užduotis turi turėti pavadinimą, kurį žino ir supranta visi komandos nariai. Priklausomai nuo užduoties ir proceso, papildomai gali būti rodoma ir tokia užduočių informacija kaip [7]:

- Sukūrimo data
- Galutinis terminas
- Užduoties kūrėjas
- Prioritetas
- Užduoties tipas
- Aprašymas
- Pastabos
- Apibrėžtumas ir reikalavimai
- Istorija

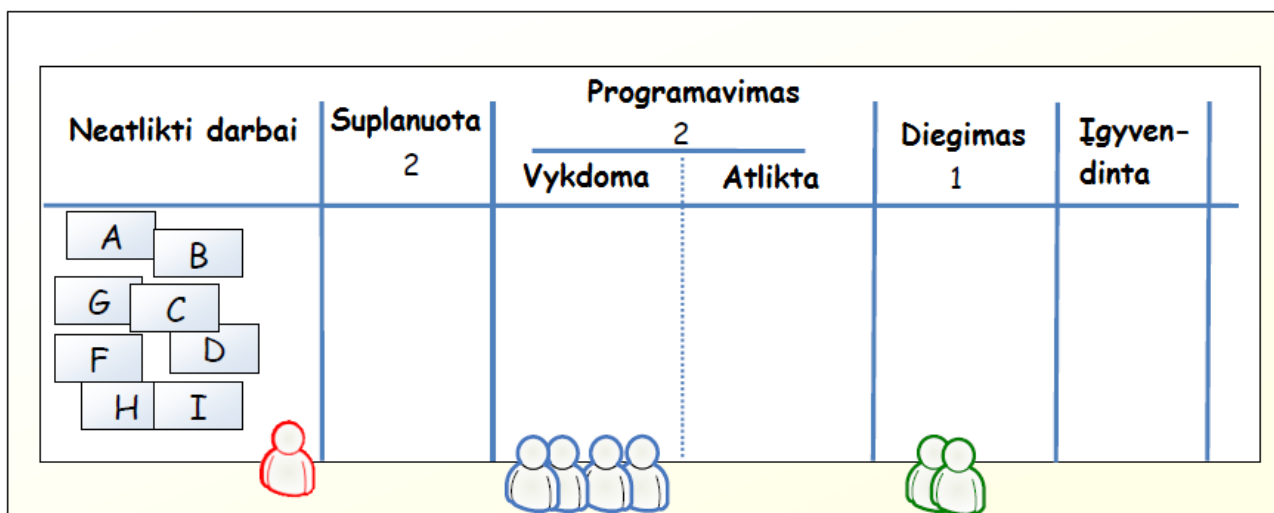
Taip pat vertėtų paminėti, kad nepriklausomai nuo to, jog komanda yra sudaryta iš vieno ar daugiau žmonių, *Kanban* lentoje vis tiek turėtų būti rodoma, kas dirba prie vykdomų užduočių.

Kad *Kanban* metodologijos naudojimas taptų dar naudingesnis, reikėtų nepamiršti ir proceso tėkmės vertinimo bei pagerinimo. Pagerinimai visuomet turėtų būti grindžiami objektyviais vertinimais ir *Kanban* metodologija – ne išimtis. Ieškoti ir taikyti gerus rodiklius, pagal kuriuos būtų galima sudaryti naudingus grafikus, atvaizduojančius sangrūdų vietas ir kitas problemas, dažnai yra gan sudėtingas žingsnis. Pavyzdžiui, sangrūdos, kai *Kanban* lenta yra pilnai sudaryta ir užpildyta, proceso eiga įsibėgėjusi, jas galima pamatyti vos tik pažvelgus į lentą, tačiau periodiškai sudarinėjant grafikus ir fiksuojant visus pokyčius, galima stebėti ar proceso efektyvumas didėja, jei didėja, tuomet, kaip sparčiai, o, galbūt, niekas nesikeičia. Visa grafikų sudarinėjimo, proceso tėkmės vertinimo, jo pagerinimo ir apskritai *Kanban* metodologijos esmė yra kiek įmanoma palengvinti problemų, kurios trukdo efektyvumo didinimui, paieškas, kad būtų galima atlikti kaip įmanoma daugiau darbų per kuo trumpesnę laiko tarpą. Ankstyvas problemos pastebėjimas padidina šansus, kad sprendimas šiai problemai spręsti, bus surastas dar prieš susikaupiant didelei krūvai iš dalies padarytų darbų. Didelis kiekis neužbaigtų darbų mažina efektyvumą, kadangi žmonės pradeda šokinėti nuo vienos užduoties, prie kitos, taip prarasdami aiškų supratimą, ką reikia daryti toliau ir kas yra svarbiausia, o visa tai didina viso darbo krūvio sudėtingumą [6].

Kanban – tai tarsi “traukimo” metodologija. Šis terminas yra kilęs iš idėjos, kad vienas proceso etapas tarytum traukia darbą iš prieš jį esamo etapo, taip duodamas signalą prieš tai buvusiam etapui, jog pastarajam galima atlikti naują užduotį. Šis principas, limituodamas neužbaigtus darbus yra visiškai priešingas ir prieštarauja “stūmimo” principui, kur kiekviename etape darbą stengiamasi padaryti kiek įmanoma greičiau, o tuomet nustumti jį į kitą etapą, nekreipiant dėmesio į tai, kiek pradėtų darbų jau yra susikaupę sekančiame etape. Bendrai kalbant, “stūmimas” reiškia, kad kai kas nors yra pasirengęs atlikti užduotį, yra žiūrima į lentą, kad pamatyti, ką tuo metu reikia padaryti, parinkta užduotis, atsižvelgiant į pradėtų darbų apribojimo limitą, yra perkeliama į atitinkamą stulpelį. Užduotis tampa to žmogaus, kuris pradėjo ją vykdyti, atsakomybe iki tol, kol ji nėra perstumiami į kitą stulpelį [7].

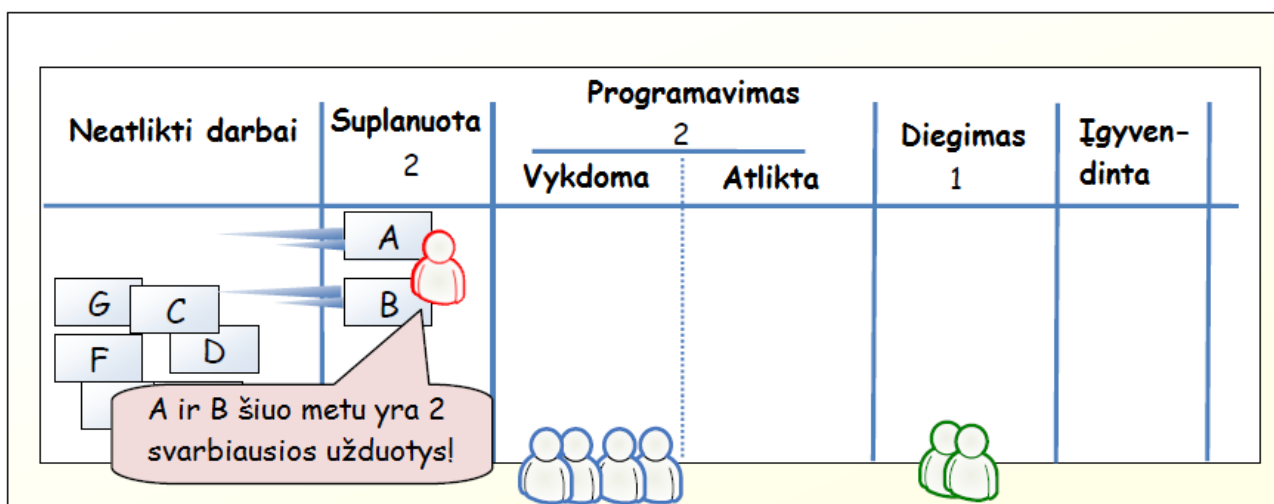
Kad viskas atrodytų kuo aiškiau ir būtų kuo lengviau suprasti, toliau yra pateikiamas pavyzdys kaip *Knaban* procesas atrodo vaizdžiai.

Pavyzdys yra ganėtinai primitivus, bet labai aiškiai atvaizduojantis esmines *Kanban* proceso vietas. Šiuo atveju tai yra nedidelė komanda (12 pav.), kurią sudaro 7 asmenys: 1 žmogus, atsakingas už reikalavimus, paprastai tai būna verslo analitikas arba produkto savininkas, taip pat 4 programuotojai ir 2 diegimo inžinieriai. *Kanban* lenta yra sudaryta, pradėtų darbų apribojimo limitai nustatyti, tad belieka pradėti skirstyti užduotis ir jas įgyvendinti.



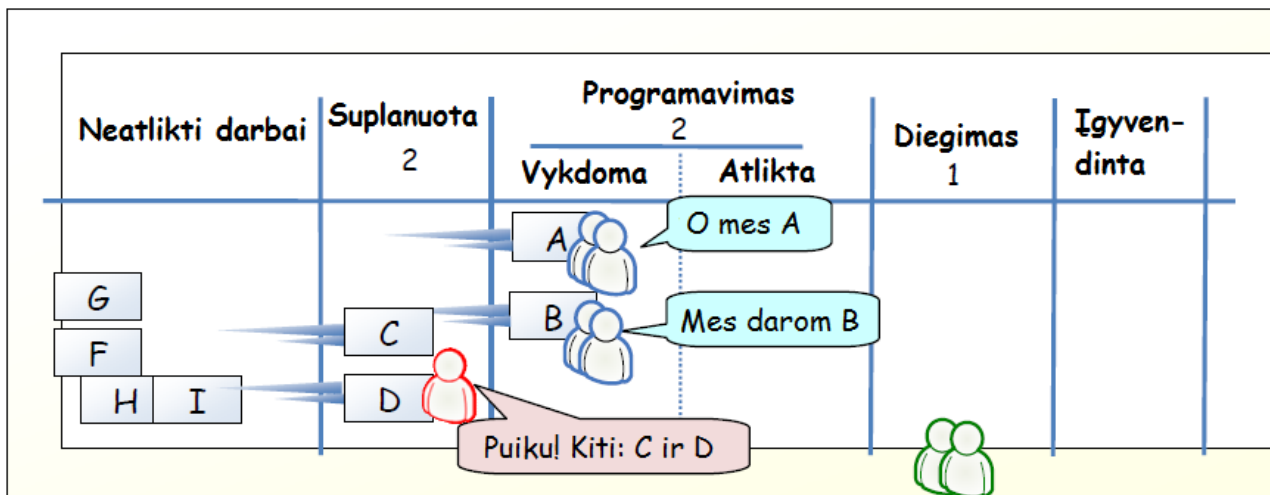
12 pav. *Kanban* lenta ir komanda [1]

Žmogus atsakingas už reikalavimus, nusprendžia, kurios užduotys yra svarbiausios, atkreipdamas dėmesį į prioritetą, ir dvi iš jų *Kanban* lentoje perstumia į suplanuotų užduočių stulpelį. Perstumiamos yra tik dvi užduotys, kadangi toks gali būti maksimalus suplanuotų užduočių skaičius.



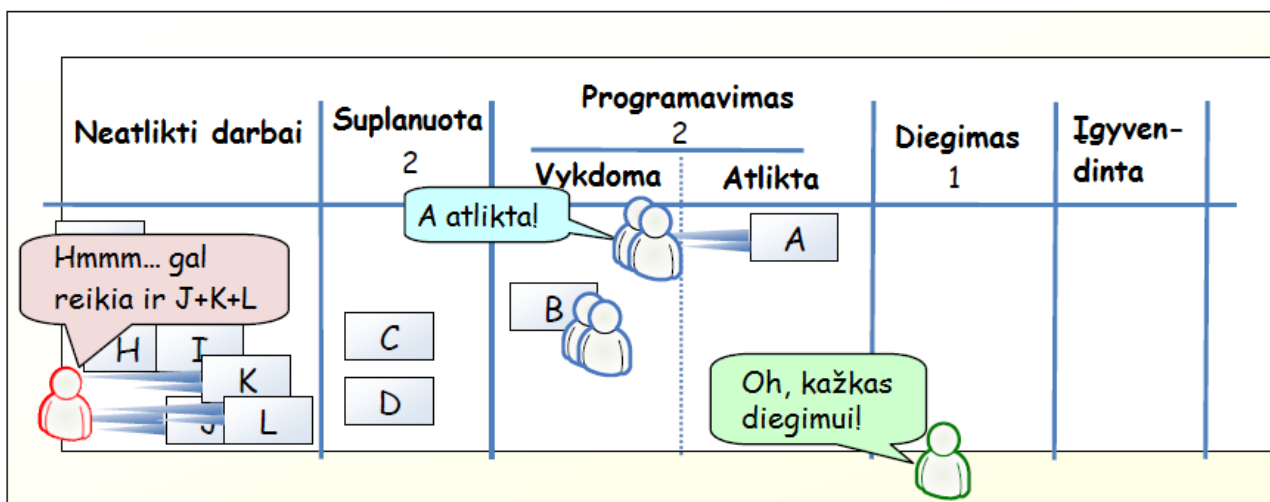
13 pav. Pagal limitą paskiriamos užduotys [1]

Šiuo atveju programavimo etapas yra suskirstytas į vykdomas ir atliktas užduotis, o bendras limitas yra 2, kadangi tiek atliktų, tiek vykdomų užduočių nėra, todėl programuoti iškart pradedamos abi suplanuotos užduotys. Tuo tarpu, atlaisvėjus suplanuotų užduočių stulpeliui iškart yra priskiriamos dar dvi, pagal prioritetą atrinktos, užduotys, kurias bus galima atlikti.



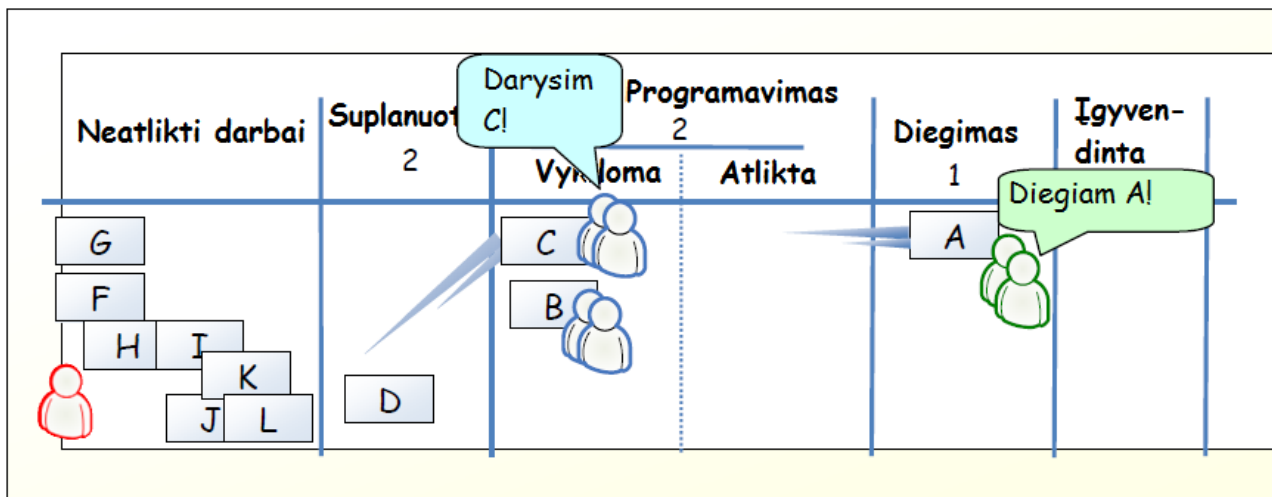
14 pav. Programuojamos ir toliau skirstomos užduotys [1]

Viena programuotojų pora baigia savo užduotį „A“ ir perkelia ją į atliktų užduočių stulpelį, tuo tarpu, kita programuotojų pora vis dar programuoja savo užduotį „B“ ir bendras užduočių skaičius programavimo etape yra 2, todėl savo užduotį baigusi programuotojų pora negali imti sekančios užduoties, kol diegimo inžinieriai nepasiims ir nepradės daryti užduoties, kuri yra atliktų užduočių stulpelyje.



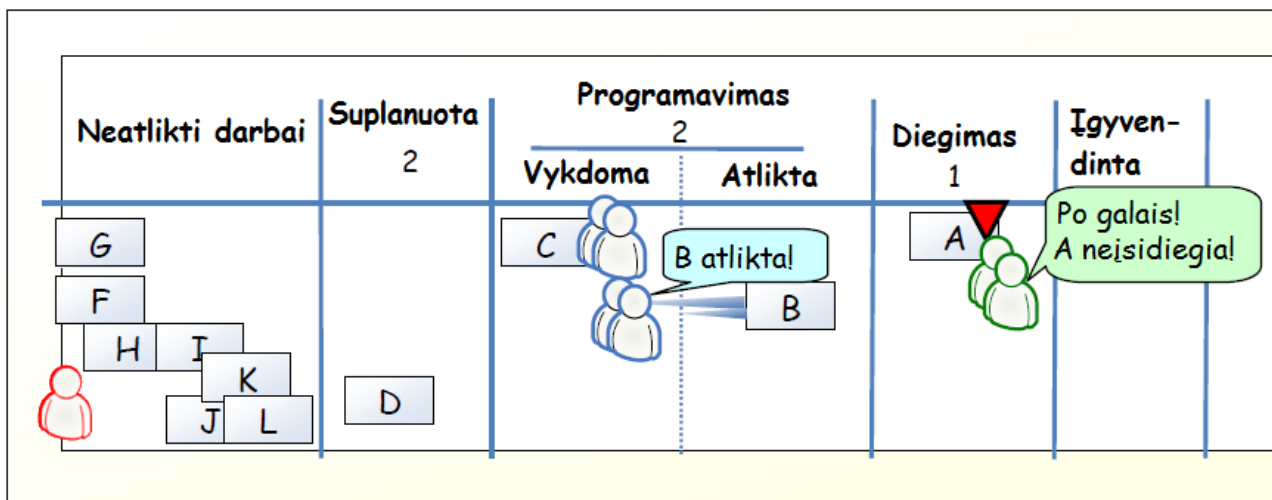
15 pav. Atlikta užduotis atiduodama diegimui [1]

Diegimo inžinieriams patraukus užduotį „A“ iš atliktų užduočių stulpelio ir pradėjus ją diegti, vienu vienetu atlaisvėja užduočių kiekis, esantis programavimo etape, todėl be užduoties esanti programuotojų komanda, gali imti sekančią užduotį ir pradėti ją programuoti, o žmogus, atsakingas už reikalavimus, nuspręsti, kuri užduotis turi būti suplanuota sekanti.



16 pav. Programuojamos ir diegiamos užduotys [1]

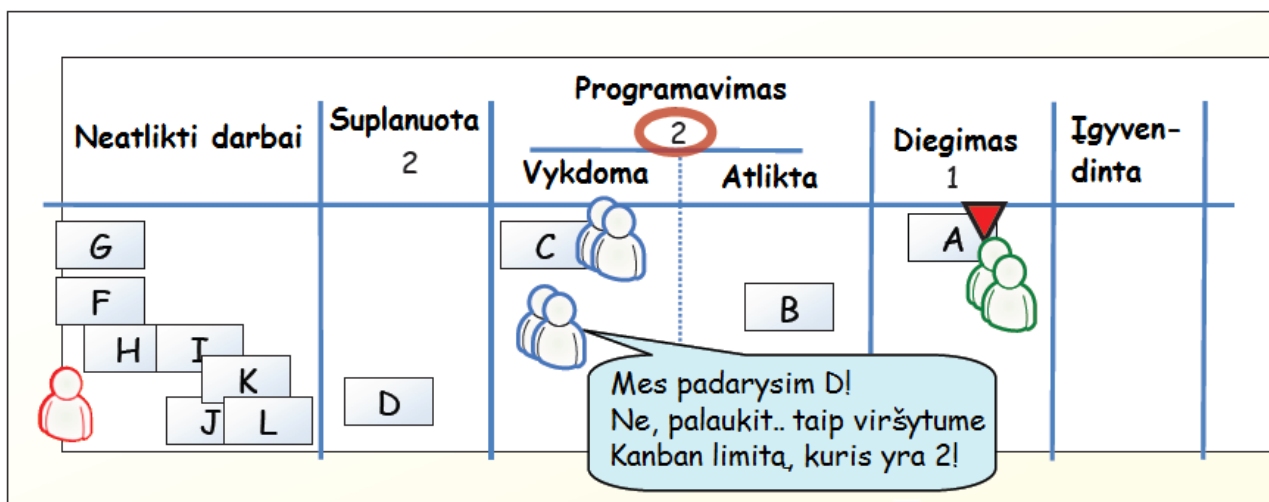
Kita programuotojų pora baigia savo užduotį „B“ ir perkelia ją prie atliktų užduočių stulpelio, tuo tarpu, diegimo inžinieriai susiduria su netikėtai iškilusiomis problemomis ir negali sėkmingai įdiegti užduoties „A“.



17 pav. Susiduriama su problemomis diegiant užduotį [1]

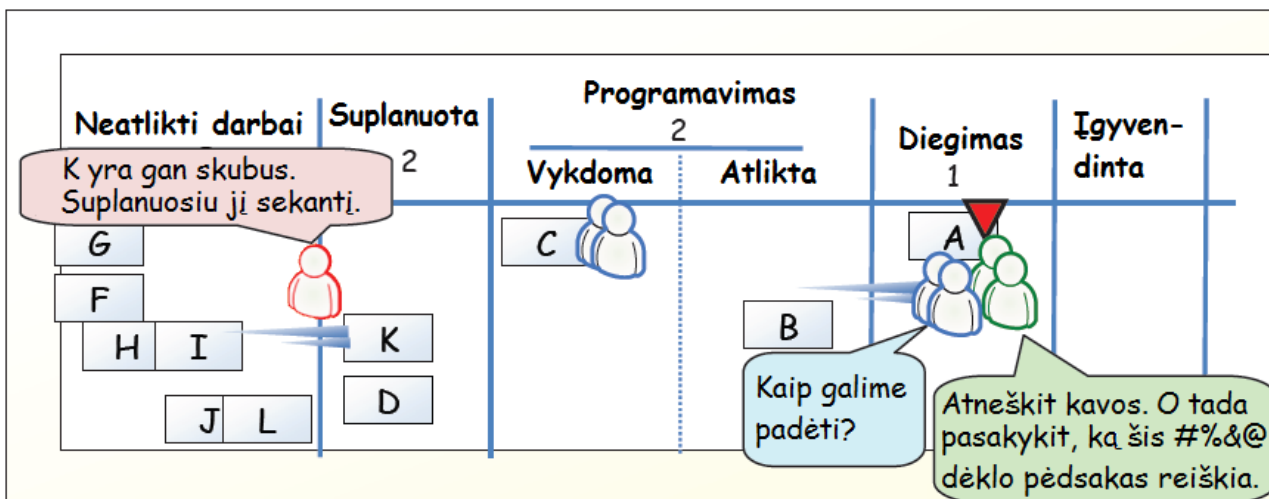
Programuotojų komanda, įvykdžiusi savo užduotį „B“, norėtų imti daryti sekančią užduotį, tačiau negali to daryti, kadangi, taip viršytų bendrą programavimo etapo limitą, kuris yra 2. Tuo

tarpu diegimo inžinieriai vis dar stengiasi įdiegti užduotį „A“ ir dėl šios priežasties negali pradėti diegti užduoties „B“, kadangi diegimo etape maksimalus pradėtų užduočių limitas yra tik viena užduotis.



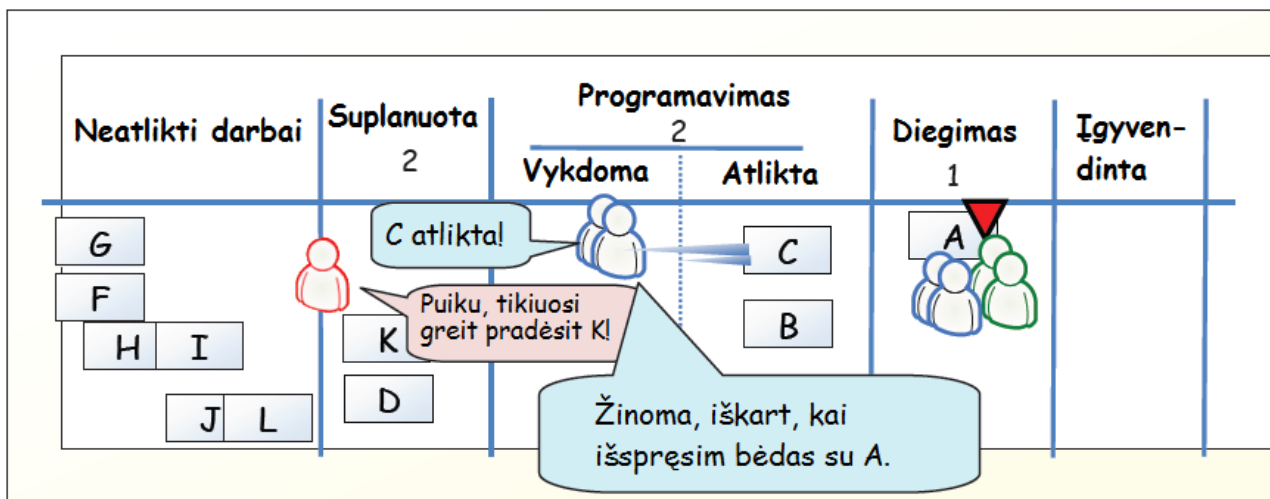
18 pav. Pasiektas maksimalus programavimo etapo limitas [1]

Pagal Kanban taisykles, programuotojų pora, nebeturinti kito pasirinkimo, kaip tik laukti, kol sumažės užduočių programavimo etape, eina padėti diegimo inžinieriams spręsti susiklosčiusių problemų, kad šios būtų kaip įmanoma greičiau pašalintos ir toliau būtų galima sėkmingai vykdyti kitas užduotis. Tuo tarpu, žmogus, atsakingas už reikalavimus, suplanuoja dar vieną užduotį, taip pasiekdamas suplanuotų užduočių apribojimo limitą.



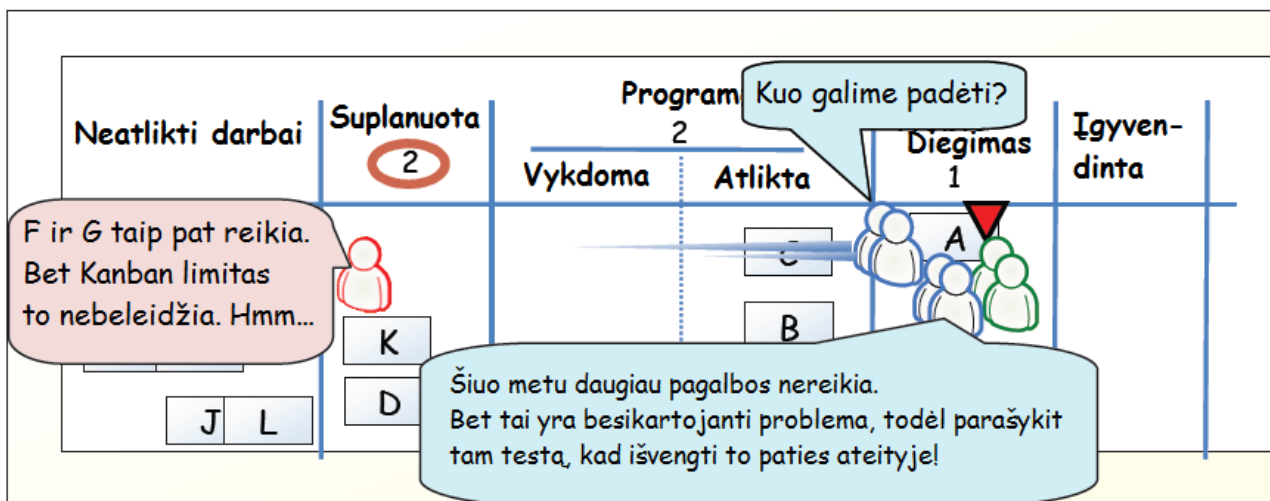
19 pav. Programuotojai padeda diegimo inžinieriams [1]

Ir kita programuotojų pora baigia savo užduotį, perstumia ją į atliktų užduočių stulpelį, tačiau kitos užduoties jie pradėti negali, kadangi diegimo inžinieriai, kartu su kita programuotojų pora vis dar bando išspręsti problemas, kurios iškilo diegiant „A“ užduotį. Reikalavimus skirstančiam žmogui reikia, kad suplanuotos užduotys būtų pradėtos kuo greičiau, tačiau to negalima padaryti, kol nebus išspręstos diegimo problemos.



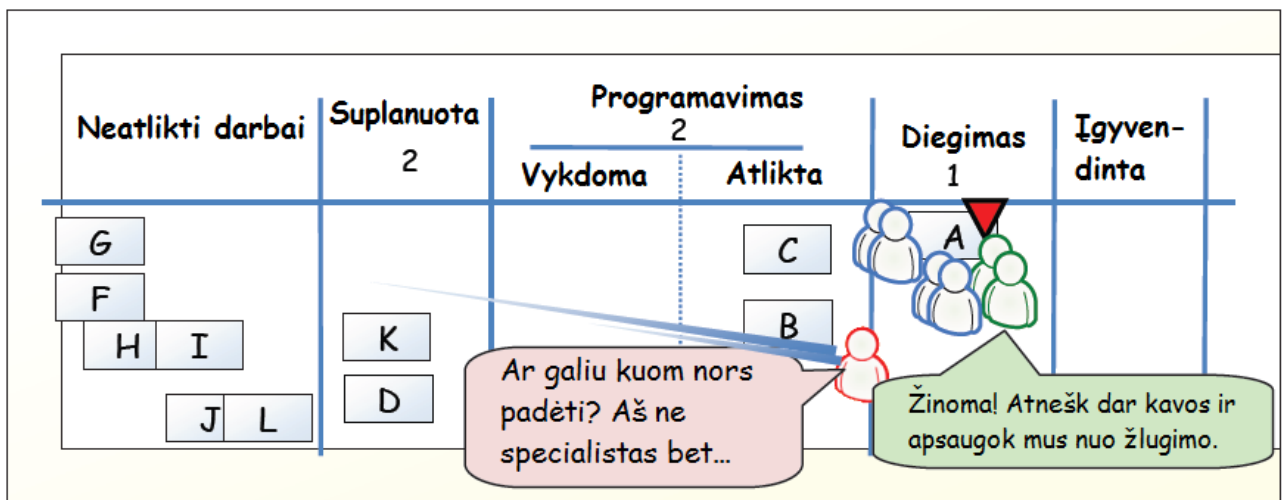
20 pav. Programuotojai padeda diegimo inžinieriams [1]

Kita programuotojų pora taip pat prisijungia prie visų, besistengiančių išspręsti diegimo problemas, bendromis pastangomis, stengiamasi surasti sprendimus, kurie viską išspręstų ir užtikrintų, kad tokios problemos nebepasikartotų ateityje. Tuo tarpu, nė vienam programuotojui neprogramuojant, suplanuotos užduotys nėra vykdomos, o jau reikia planuoti naujas, tačiau to padaryti negalima dėl nustatyto limito.



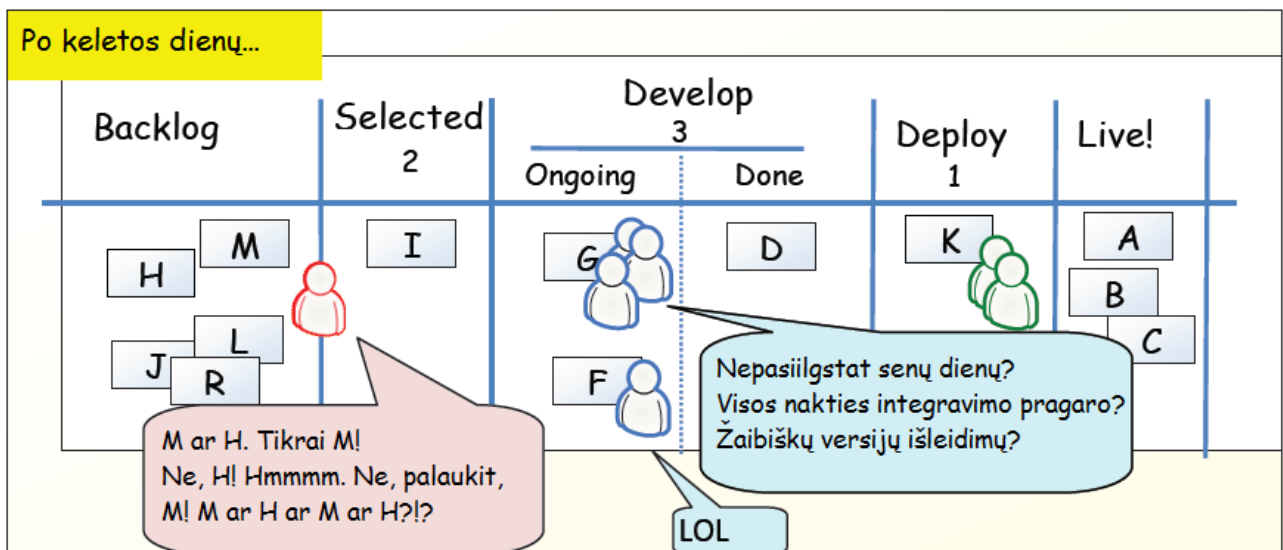
21 pav. Programuotojai padeda diegimo inžinieriams [1]

Žmogui, atsakingam už reikalavimus, nebelia nieko kito, kaip tik pasiteirauti, ar jis gali būti kuom nors naudingas, kadangi nebėra prasmės dėti prioritetų užduotims, kol nė viena iš jų nėra programuojama.



22 pav. Visa komanda stengiasi išspręsti diegimo problemą [1]

Praėjus keletai dienų ir išsprendus diegimo problemas, viskas stoja į savo vietas. Diegimo inžinieriai sėkmingai diegia naujas užduotis, programuotojai programuoja, o reikalavimų žmogus bendrauja su klientais ir sprendžia, kurios gi užduotys yra svarbesnės.



23 pav. Sėkminga proceso eiga išsprendus problemas [1]

Šiuo pavyzdžiu buvo pavaizduota kaip vyksta Kanban procesas, kas nutinka tuomet, kai susiduriama su kažkokiomis problemomis, šiuo atveju buvo diegimo problema, jei tokios problemos pradėtų kartotis periodiškai, tuomet reikėtų pradėti galvoti ne tik apie tai kaip išspręsti

diegimo problemas, bet ir apie tai kaip padaryti, jog jos daugiau nebesikartotų, gal tai diegimo inžinierių patirties stoka, gal programuotojų netinkamai suprogramuojamos užduotys, o gal testuotojų trūkumas, kurie, galbūt, padėtų surasti problemas, kurias ištaisius, nebebūtų problemų diegiant užduotis. Pradėjus naudoti *Kanban* metodologiją išryškėja esminės egzistuojančios problemos, kurias išsprendus, *Kanban* tampa puikiai tinkama efektyviam darbui užtikrinti.

2.3 *Kanban* metodologijos privalumai

Kaip ir daugelis žmonijos išradimų, taip ir *Kanban* metodologija, turbūt, nebūtų naudojama, jei neturėtų akivaizdžių privalumų [18]:

- Vienas pagrindinių *Kanban* metodologijos privalumų yra darbo efektyvumo didinimas apribojant pradėtų darbų kiekį. Naudojant *Kanban* metodologiją, didelis dėmesys yra skiriamas darbams, kurie yra pradėti ir jų apribojimui. Nustatyti apribojimai, leidžia sutaupyti laiko, kadangi stengiamasi susikonscentruoti ties vienu darbu ir nesimėtoma tarp kelių ar keliolikos darbų.
- Silpnų proceso vietų nustatymas. Stebint *Kanban* lentą gan greitai galima nustatyti, kurioje vietoje susidaro sangrūdos, kodėl kartais vėluojama atlikti užduotis, ar kuris darbuotojas nuolat nespėja atlikti darbų ir panašiai. Kuo problemos anksčiau pastebimos, tuo greičiau jas galima išspręsti, taip sutaupant laiko ir išvengiant didesnių problemų.
- Proceso optimizavimas, sudarinėjant grafikus. Grafikai padeda optimizuoti procesą laikui bėgant, kadangi galima atlikti įvairias analizes, kurios padeda nustatyti problemas, kurios tiesiog pažvelgus į *Kanban* lentą nėra matomos.
- Skatinamas darbuotojų atsakingumas, kadangi kiekvienas darbuotojas, tuo metu, kai atlieka jam priskirtą užduotį, yra už ją atsakingas, kad pastaroji būtų atlikta laiku ir pasiektų sekantį žingsnį.

2.4 *Kanban* metodologijos trūkumai

Sukurti kažką tobulo ir be jokių trūkumų yra itin sunki užduotis, net tobuliausiems ir geriausiems žmonijos kūriniams kartais lemta suklysti. Ne išimtis ir *Kanban* metodologija, ši metodologija turi tikrai svarių privalumų, kodėl ją vertėtų naudoti, tačiau turi ir keletą trūkumų bei vietų, kurias būtų galima šiek tiek patobulinti artėjant tobulumo link [21]:

- Praeitame skyriuje išanalizavus *Scrum* metodologiją, kaip vieną iš *Kanban* metodologijos trūkumų būtų galima paminėti susirinkimų stygių. *Kanban* metodologijoje nėra numatyta jokių planavimo susirinkimų, darbus planuoja vienas žmogus, tuo tarpu, žmonės, kurie turi įgyvendinti tuos darbus, prieš pradėdami atlikti kažkokią užduotį, dažnai ją mato pirmą kartą. Taip yra gaišamas kai kas, kol yra išsiaiškinami visi reikalavimai.
- Ne tokia efektyvi, kai reikia atlikti užduotis, kurios yra priklausomos ir nuo kitų projektų ar komandų. Pradėjus užduotį, kuri negalės būti pilnai įgyvendinta, kol kita komanda neatliks savo dalies, gali susidaryti sangrūda. Jei viena komanda pasiekia pradėtų darbų apribojimo limitą, tarkime, programavimo etape, ir visos užduotys, arba didžioji dalis, yra priklausomos nuo kitos komandos, o pastaroji taip pat yra pasiekusi savo pradėtų darbų apribojimo limitą programavimo etape, bet jame nėra nė vienos užduoties, kuri būtų reikalinga pirmajai komandai, tokiu atveju pirmoji komanda neturi kito pasirinkimo, kaip tik laukti, kol antroji komanda pradės atlikti jiems reikalingas užduotis, arba viršyti savo pradėtų darbų apribojimo limitą.
- Naudojant šį projektą griežtai pagal taisykles, kad apribojimo limitai nebūtų viršijami arba su labai žemais apribojimo limitais, siekiant procesą padaryti kuo efektyvesnį, reikalinga itin universalių žmonių komanda. Dauguma programuotojų yra linkę tiesiog programuoti netestuodami, todėl, jei atsiranda sangrūdos ties testavimo etapu, nemaža tikimybė, jog testuotojams į pagalbą atėję programuotojai testuos ne taip atsakingai, o tai padidina riziką išleisti ne tokią kokybišką programinės įrangos versiją. Tuo tarpu, jei sangrūdos atsiranda programavimo etape, vargu ar atsiras daug testuotojų, kurie moka programuoti, o jei tokių yra, dažniausiai, po kurio laiko, jie tampa programuotojais.

2.5 Išvados

Išnagrinėjus *Kanban* metodologiją, jos privalumus be trūkumus, nustatyta, kad ši metodologija yra orientuota į darbo efektyvumo didinimą, siekiant apriboti pradėtų ir neužbaigtų darbų kiekį, bei proceso optimizavimą. Metodologija turi keletą trūkumų, tačiau juos galima gan lengvai išspręsti pagal projekto poreikius.

3. Darbo efektyvumo tyrimas, naudojant *Scrum* ir *Kanban* metodologijas

3.1 Įvadas

Teoriškai kiekvienas projektas turėtų būti sėkmingai įgyvendinamas naudojant tiek *Scrum*, tiek *Kanban* metodologijas. Darbai turėtų būti atliekami laiku, o procesai optimizuoti taip, kad darbo efektyvumas būtų aukštas, tačiau praktika nuo teorijos gali šiek tiek skirtis, o metodologijos būti viena už kitą pranašesnės vienoje ar kitoje vietoje. Remiantis vien teorija būtų sunku apsispręsti, kurią iš šių metodologijų vertėtų naudoti, norint pradėti naują projektą, todėl prieš pasirenkant reikėtų atsižvelgti ir į praktinę metodologijų analizę.

Šio skyriaus tikslas – atlikti darbo efektyvumo tyrimą, naudojant tiek *Scrum*, tiek *Kanban* metodologijas bei atlikti šių metodologijų palyginamąją analizę, taip nustatant, kuri iš šių metodologijų yra efektyvesnė.

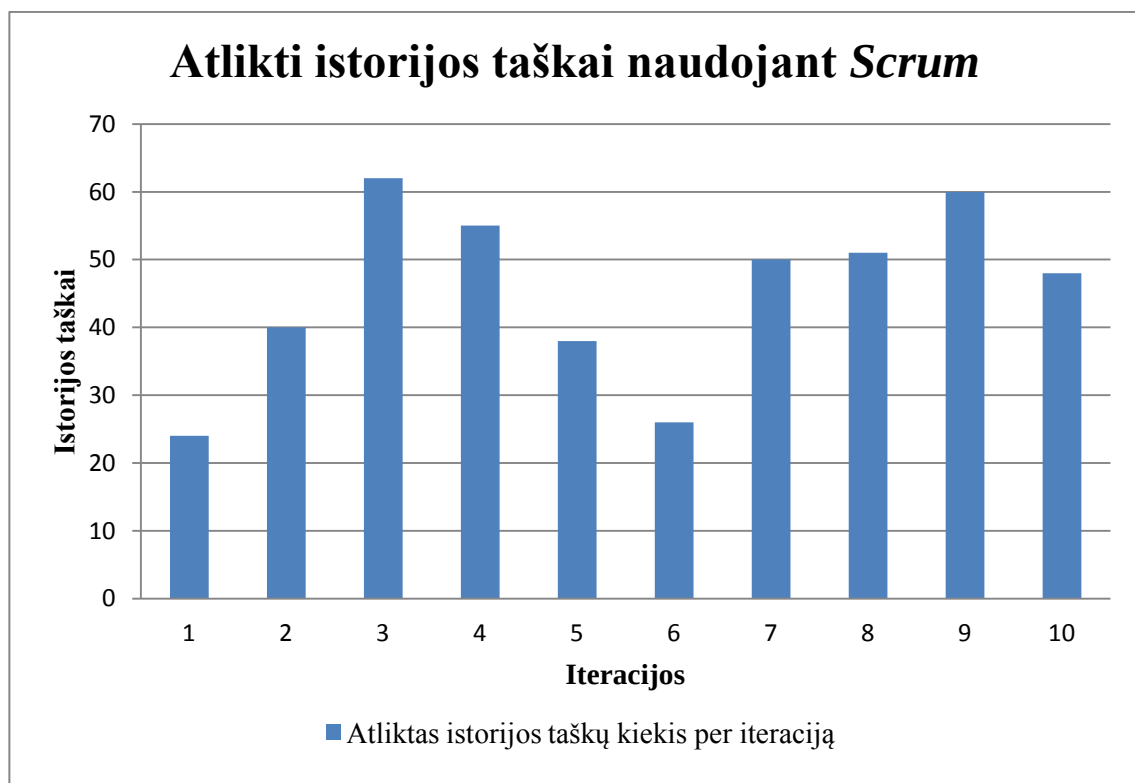
3.2 Darbo efektyvumas naudojant *Scrum*

Norint kuo tiksliau atlikti *Scrum* metodologijos efektyvumo tyrimą, buvo naudojami jau keletą metų egzistuojančio projekto duomenys. Tyrimas buvo atliekamas naudojant duomenis apie praleistą laiką įgyvendinant vienas ar kitas užduotis, kurios buvo įvertintos istorijos taškais (angl. *Story points*).

Istorijos taškai – tai priemonė, nustatyti reikiamą pastangų kiekį, norint įgyvendinti užduotis. Kitaip tariant, tai tiesiog skaičius, kuris nusako, kokio sudėtingumo užduotis yra. Dažniausiai yra naudojama fibonačio seka (angl. *Fibonacci sequence*).

Duomenys, tyrimui atlikti, buvo paimti paskutiniųjų dešimties iteracijų laikotarpio, kai dar buvo dirbama pagal *Scrum* metodologiją. Viena iteracija trunka tris savaites, taigi bendras laikotarpis, kurio metu buvo renkami duomenys, yra apie 7 mėnesiai ir 2 savaitės. Per šį laikotarpį buvo atlikta nemažai užduočių, kurios visos buvo įvertintos 454 istorijos taškais. Visi šie atlikti istorijos taškai, kurie apibrėžia atliktų užduočių kiekį, yra atvaizduojami 24 paveiksle pateiktame grafike, kuriame jie yra išskirstomi pagal tai, kiek jų buvo padaryta kiekvienos iteracijos metu.

Horizontaliojoje ašyje yra atvaizduojamos visos iteracijos, o vertikaliojoje ašyje atvaizduojami istorijos taškai.



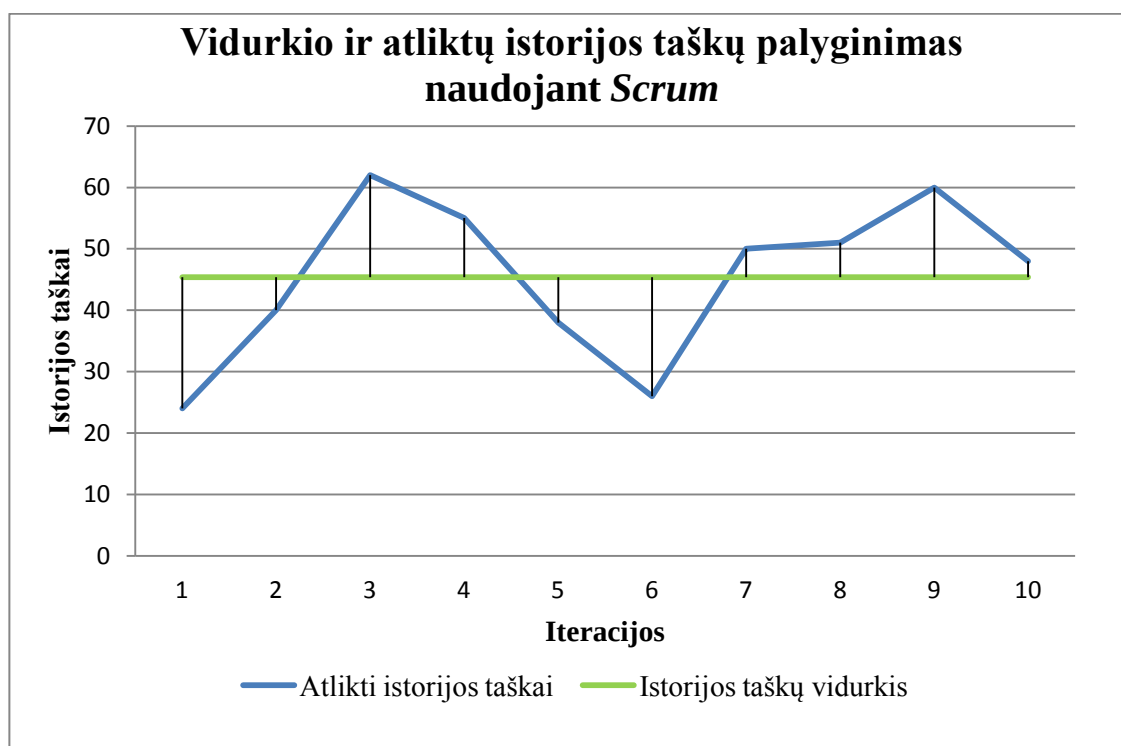
24 pav. Atlikti istorijos taškai per iteracijas naudojant *Scrum*

Iš grafiko matyti, jog yra keturios iteracijos (1, 2, 5 ir 6), gan ryškiai išsiskiriančios iš visų atvaizduotų. Aiškiai matyti, jog tų iteracijų metu atliktas istorijos taškų kiekis yra mažesnis, nei likusių šešių. Grafikas atrodo tarsi banga, pradedant mažiausiu atliktų istorijos taškų kiekiu (pirma iteracija), toliau atliktas kiekis po truputį kyla, o pasiekęs trečiąją iteraciją, nuo ketvirtosios atliekamas istorijos taškų kiekis pradeda mažėti, kol nuo septintosios vėl pradeda didėti. Kadangi šiame grafike atvaizduota dešimtoji iteracija yra paskutinė iteracija, kai projektas dar dirbo pagal *Scrum* metodologiją, toliau šio grafiko pratęsti tiesiog nėra galimybės, bet jei grafikas būtų pratęstas, ties vienuolikta ar dvylikta iteracija atliktų istorijos taškų skaičius tikrai turėtų būti mažesnis ir dar kelias iteracijas mažėti. Taip yra todėl, kad *Scrum* metodologija reikalauja užduotis planuoti iteracijomis ir visas suplanuotas užduotis privaloma atlikti per tą iteraciją, kuriai jos buvo suplanuotos. Taigi, norint atlikti užduotis, kurias įgyvendinti užtrunkama ilgiau nei tris savaites, pradžioje jos tiesiog nėra planuojamos nė vienai iteracijai, bet pradedamos programuoti, o vėliau, sekančioje ar dar sekančioje iteracijoje, suplanuojamos ir užbaigiamos. Taip reikia elgtis todėl, kad

užduotis negali būti suplanuota ir neužbaigta per tris savaites. Trijų savaitių iteracija šiame projekte naudojant *Scrum* metodologiją atrodo maždaug taip:

- Pirmoji savaitė: Programuotojai pradeda programuoti užduotis, kurios yra suplanuotos dabartinei iteracijai arba programuoja užduotis, kurios bus planuojamos sekančiai iteracijai, testuotojai rašo testus.
- Antroji savaitė: Programuotojai po truputį baigia programuoti suplanuotas užduotis arba tęsia užduočių, kurios bus suplanuotos sekančiai iteracijai, programavimą, testuotojai po truputį pradeda testuoti suplanuotas užduotis.
- Trečioji savaitė: Programuotojai pradeda programuoti užduotis, kurios yra įvertintos dideliu istorijos taškų skaičiumi arba, kurios bus planuojamos sekančioje iteracijoje.

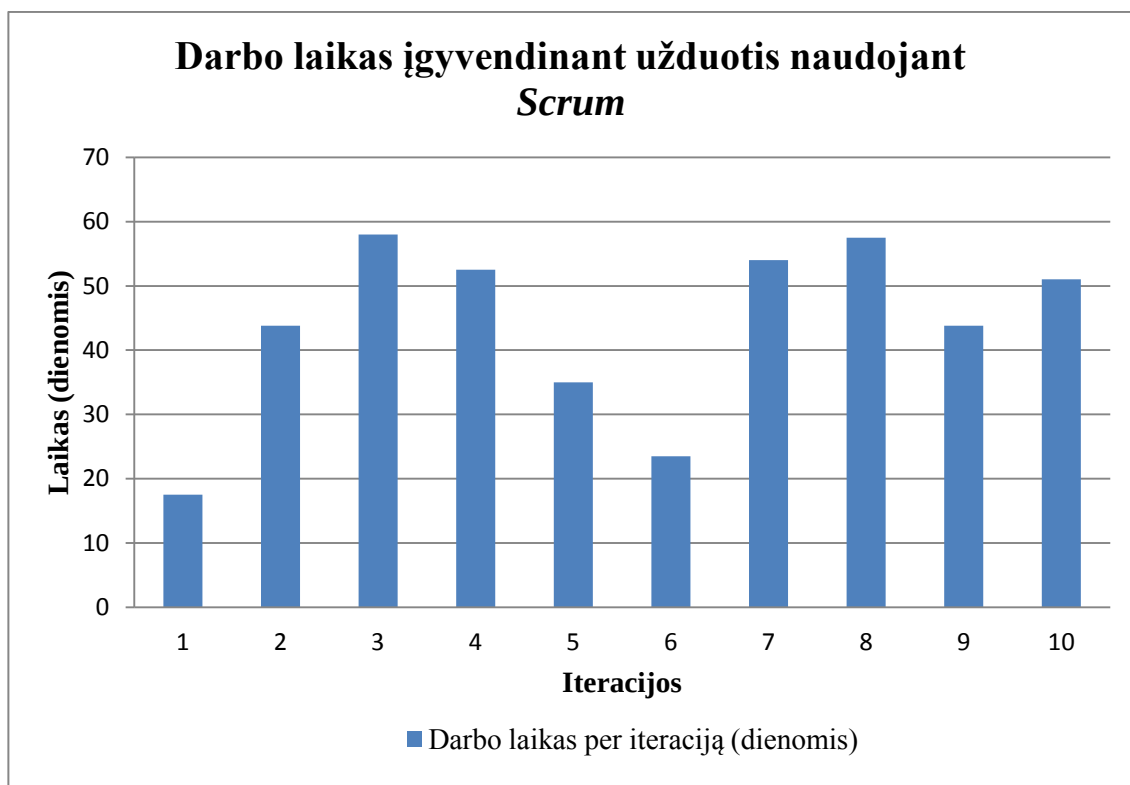
Dėl tokios darbų tėkmės ir griežto darbų atlikimo laikotarpio darbai negali būti atliekami sistemingai, kad kiekvienos iteracijos metu atliktas istorijos taškų kiekis būtų panašus (25 pav.).



25 pav. Vidurkio ir atliktų istorijos taškų palyginimas naudojant *Scrum*

25 paveiksle pavaizduotame grafike matyti, jog yra keturi ryškūs taškai (1, 3, 6 ir 9 iteracijos), kai atliktų istorijos taškų skaičius gerokai skiriasi nuo vidurkio. Iš to galima spręsti, jog tai buvo iteracijos, kuomet buvo suplanuojamos tokios užduotys, kurios buvo pradėtos įgyvendinti

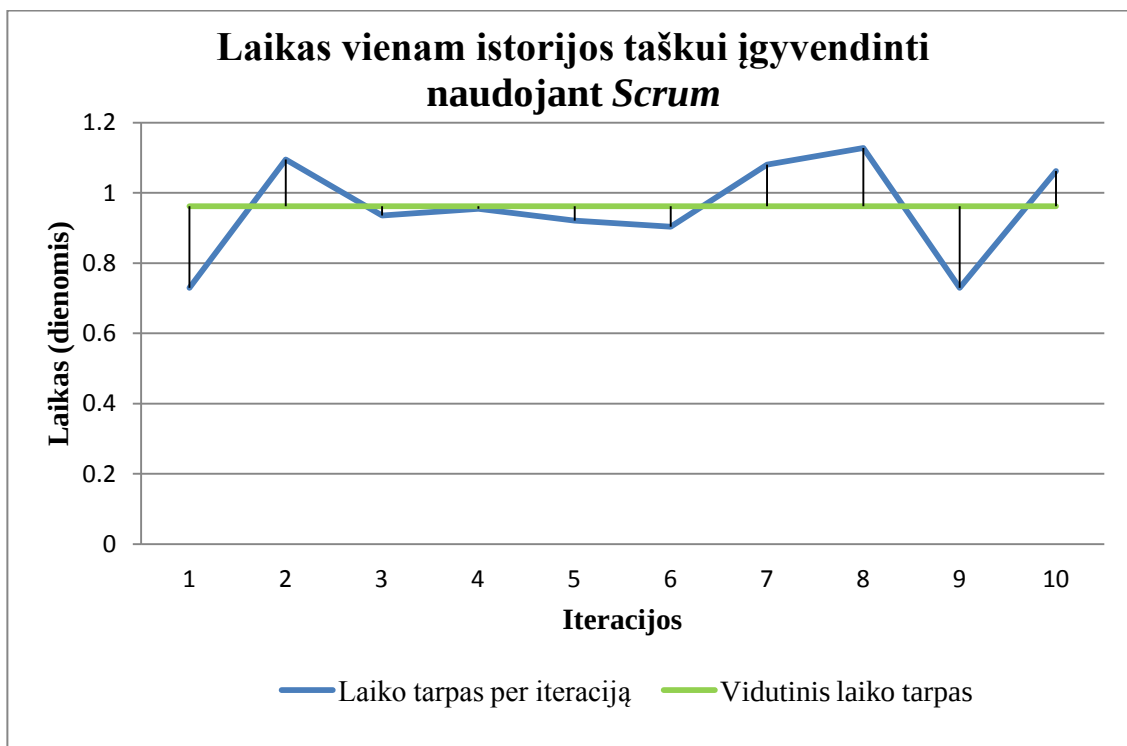
prieš tai buvusių iteracijų metu, kas pagal *Scrum* metodologiją nėra labai teisinga, o visa tai iš dalies primena *Kanban* procesui būdingą darbo tėkmę.



26 pav. Darbo laikas, praleistas įgyvendinant užduotis, naudojant *Scrum*

26 paveiksle pavaizduotas grafikas rodo, kiek laiko buvo dirbama įgyvendinant užduotis kiekvienos iteracijos laikotarpiu. Šis grafikas šiek tiek primena grafiką, pavaizduotą 24 paveiksle. Kadangi laikas, praleistas įgyvendinant užduotis, kurios bus planuojamos tik sekančiai iteracijai, negali būti priskaičiuojamas prie tos iteracijos, kurios metu tos užduotys iš tiesų yra vykdomos, todėl grafikas vėl yra „banguojantis“ ir gali atrodyti, jog vienos iteracijos metu yra praktiškai nedirbama, o kitų iteracijų metu neriamasi iš kailio, kad atlikti kuo daugiau darbų. Tokia situacija vėlgi yra todėl, kad suplanuoti darbai turi būti atliekami per tam tikrą laiką.

Sekančiame, 27 paveiksle, pavaizduota, kiek vidutiniškai laiko užtrunkama įgyvendinant vieną istorijos tašką ir kiek laiko užtrunkama konkrečios iteracijos laikotarpiu. Iš grafiko matyti, jog yra dvi (1 ir 2 iteracijos) ypač išsiskiriančios vietos ir keturios (2, 7, 8 ir 10) šiek tiek mažiau. Visi šie labiau pastebimi išsiskyrimai gali reikšti tai, jog per užduočių planavimą užduotys buvo suplanuotos nepakankamai tiksliai, todėl vienas pavyko atlikti greičiau, o kitoms prireikė daugiau laiko.



27 pav. Darbo laikas, praleistas vienam istorijos taškui įgyvendinti, naudojant *Scrum*

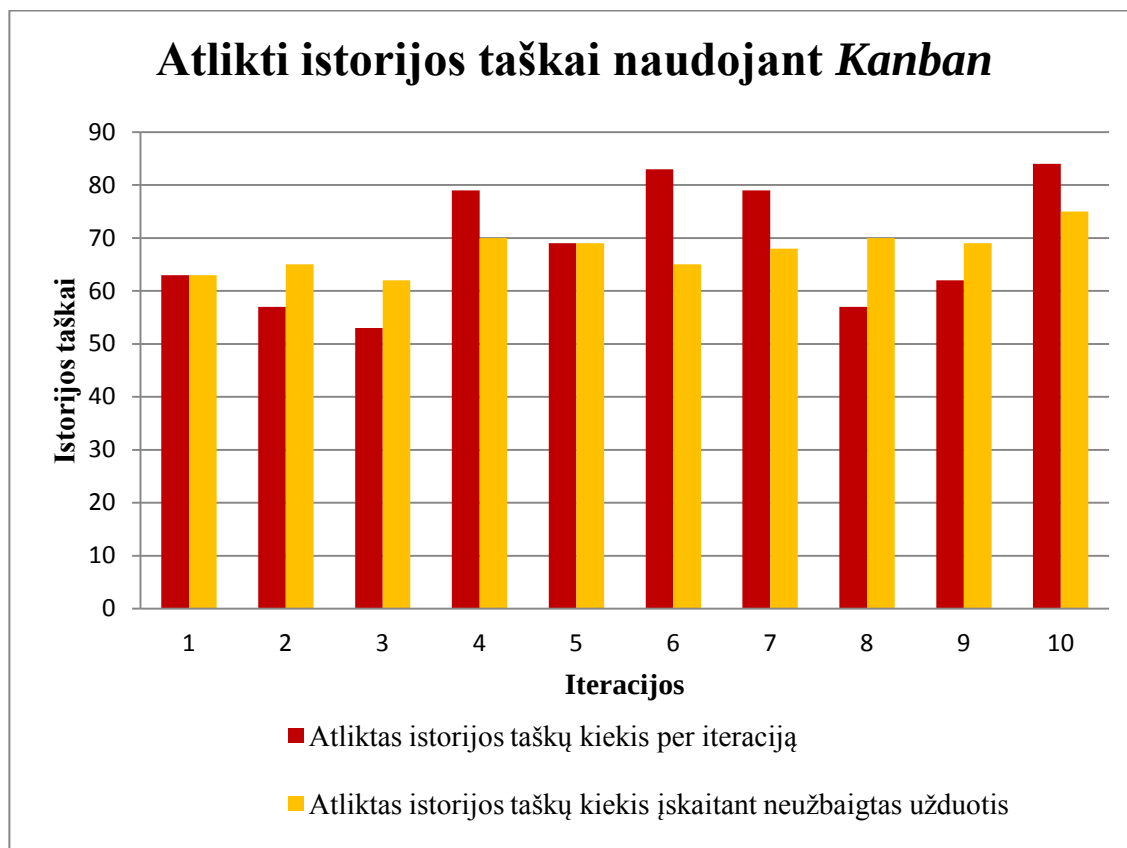
Apžvelgus visus pateiktus grafikus, nesunku pastebėti, jog naudojant *Scrum* metodologiją darbo eiga yra šiek tiek nepastovi, atliekamas darbas ir sugaišamas laikas tam darbui atlikti, skiriasi pusėje arba daugiau grafikuose pateiktų iteracijų.

3.3 Darbo efektyvumas naudojant Kanban

Norint kuo tiksliau atlikti *Kanban* metodologijos efektyvumo tyrimą, buvo naudojami taip pat to paties projekto, kuris prieš nepilnus metus dirbo vien tik pagal *Scrum* metodologiją, duomenys.

Duomenys, tyrimui atlikti, buvo paimti paskutiniųjų arba galima sakyti ir pirmųjų dešimties iteracijų laikotarpio, kadangi šiame projekte *Kanban* metodologija yra naudojama būtent apie 7 mėnesius ir 2 savaites. Kadangi prieš tai buvo naudojama *Scrum* metodologija, tai vis dar yra įprasta duomenis apie atliktas užduotis pateikti iteracijomis. Taigi, šiame tyrime viena iteracija taip pat bus prilyginama trijų savičių laikotarpiui. Per visą laikotarpį, nuo tada, kai projekte buvo pradėta naudoti *Kanban* metodologija, buvo atlikta nemažai užduočių, kurios visos buvo įvertintos 686 istorijos taškais. Visi šie atlikti istorijos taškai, kurie apibrėžia atliktų užduočių kiekį pradėjus naudoti *Kanban* metodologiją, yra atvaizduojami 28 paveiksle pateiktame grafike, kuriame jie yra

išskirstomi pagal tai, kiek jų buvo padaryta kiekvienos iteracijos metu. Horizontaliojoje ašyje yra atvaizduojamos visos iteracijos, o vertikaliojoje ašyje atvaizduojami istorijos taškai. Kadangi *Kanban* metodologija nepraktikuoja užduočių atlikimo skirstyti į laikotarpius, todėl grafike geltona spalva yra pateikiami duomenys apie tai, kiek istorijos taškų per iteracijas buvo atlikta įskaitant ir neužbaigtas užduotis.

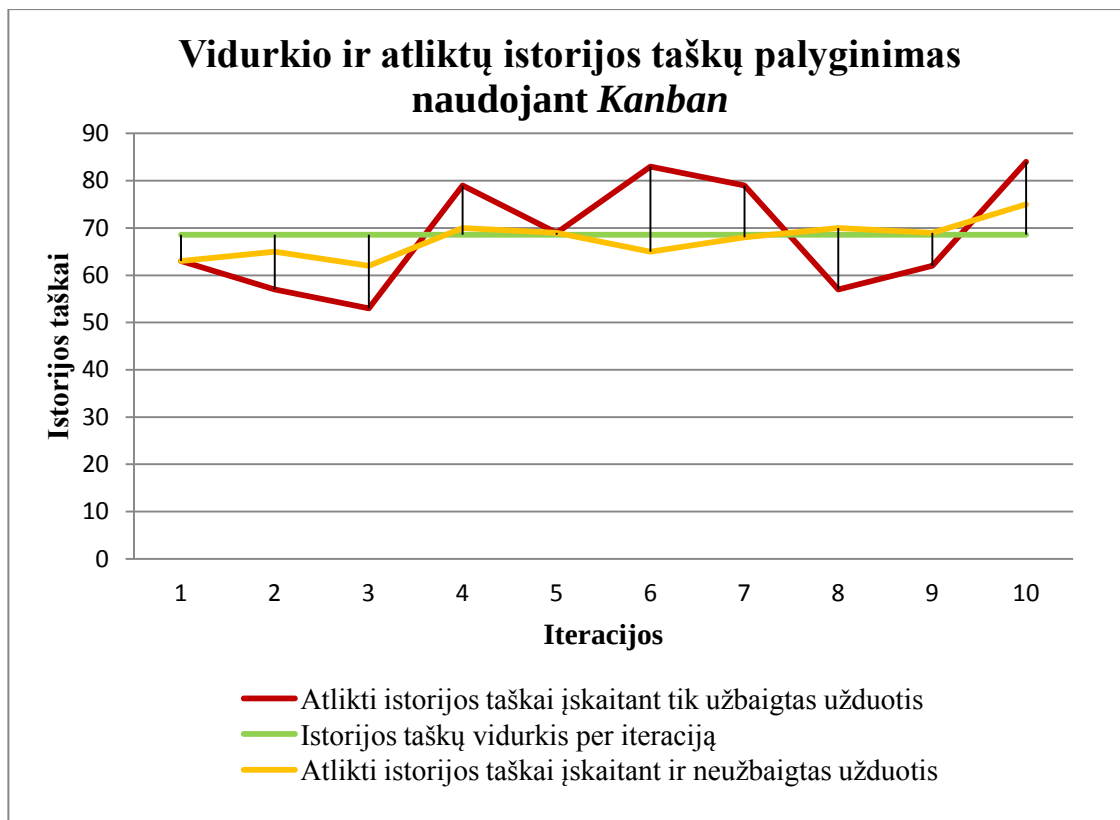


28 pav. Atlikti istorijos taškai per iteracijas naudojant *Kanban*

Pažvelgus į grafiką matyti, jog atvaizduojant istorijos taškus pagal tai, kiek jų buvo atlikta per iteraciją užbaigus visas užduotis, net ir naudojant *Kanban* metodologiją, grafikas būtų šiek tiek „banguotas“, kadangi naudojant šią metodologiją užduotys yra vykdomos nuolatos, apribojant jau pradėtų užduočių kiekį vienu metu. Tuo tarpu atkreipiant dėmesį į geltonai pavaizduotus istorijos taškus, kurie buvo atlikti per iteraciją įskaitant ir dar neužbaigtas užduotis, grafikas būtų tolygesnis, o tai parodo proceso darbų tėkmės tolygiškumą ir pastovumą.

Jei naudojant *Kanban* metodologiją duomenys apie atliekamas užduotis bus pateikiami tik apie jau užbaigtų užduočių istorijos taškus per tam tikrą laikotarpį, tai iš esmės nedaug skirsis nuo *Scrum* metodologijos grafiko, tačiau *Kanban* metodologija darbų tėkmės nespaudžia ir griežtą

darbų atlikimo laikotarpį, todėl įskaičius dar neatliktų užduočių jau atliktus istorijos taškus matyti, jog darbai gali būti atliekami sistemingai, kad kiekvienos iteracijos metu atliktas istorijos taškų kiekis yra panašus (29 pav.).

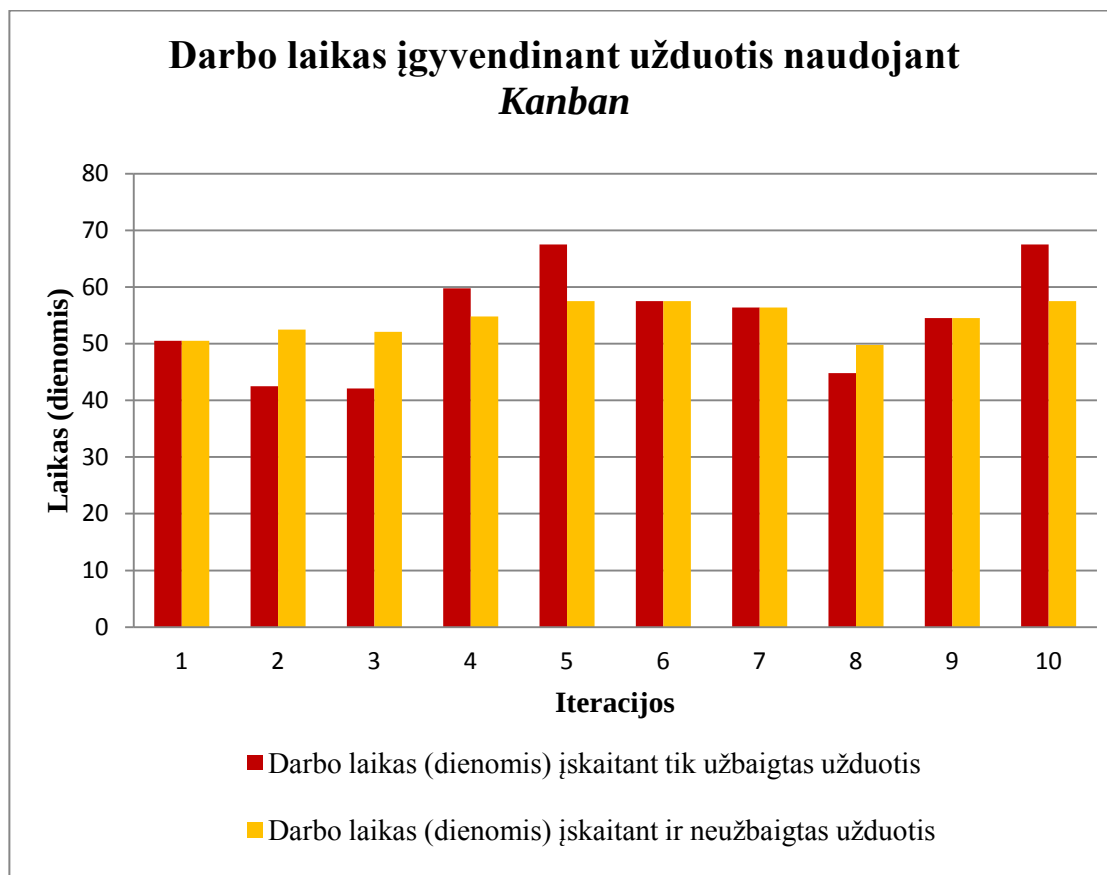


29 pav. Vidurkio ir atliktų istorijos taškų palyginimas naudojant *Kanban*

29 paveiksle pavaizduotame grafike matyti, jog beveik kiekvienos iteracijos metu atliktų istorijos taškų skaičius, įskaitant tik užbaigtas užduotis, gerokai skiriasi nuo vidurkio. Taip yra todėl, kad toks atvaizdavimas yra daugiau *Scrum* metodologijos dalis, o ne *Kanban*. Atkreipiant dėmesį į geltonos spalvos liniją, kuri atvaizduoja atliekamą istorijos taškų skaičių per iteraciją įskaitant ir neužbaigtas užduotis, matyti, jog kiekvieną iteraciją atliekamas skaičius yra kur kas arčiau vidurkio, kadangi nežiūrima vien tik į atliktas užduotis.

30 paveiksle pavaizduotas grafikas rodo, kiek laiko buvo dirbama įgyvendinant užduotis kiekvienos iteracijos laikotarpiu. Šis grafikas šiek tiek primena grafiką, pavaizduotą 28 paveiksle. Kadangi iteracijų ataskaitos vis dar yra pateikiamos pagal tai, kiek laiko buvo praleista įgyvendinti užduotims, kurios buvo užbaigtos per tą iteraciją, o ne realiai praleistas darbo laikas tos iteracijos metu, todėl grafikas vėl yra „banguojantis“ ir gali atrodyti, jog vienos iteracijos metu yra dirbama

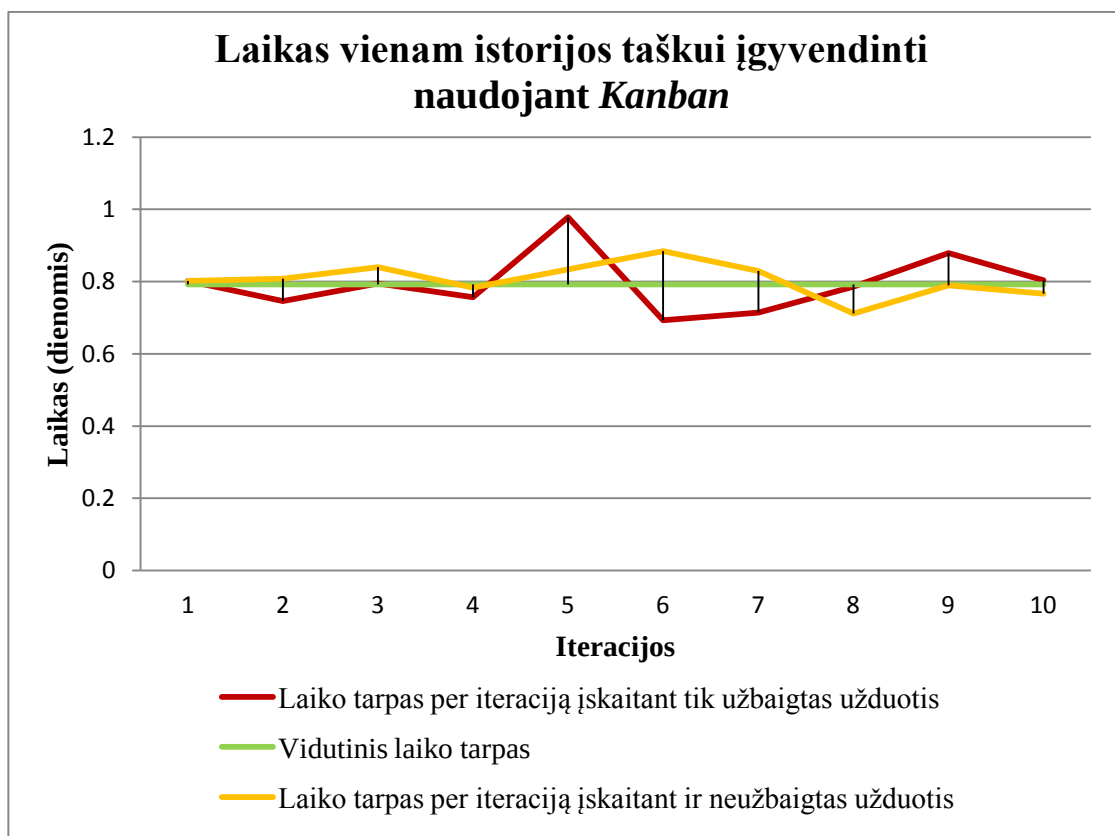
mažiau, o kitų iteracijų metu dirbama daug daugiau, kad atlikti kuo daugiau darbų. Tokia situacija vėlgi yra todėl, *Kanban* metodologija neriboja užduočių atlikimo laiko. Pažvelgus į grafike geltonai pateiktus duomenis, matyti, jog grafikas būtų kur kas tolygesnis, nei radonai atvaizduotas darbo laikas. Taip yra todėl, kad geltonai atvaizduojamas laikas yra realiai praleistas laiko tarpas per iteraciją įgyvendinant ir neužbaigtas užduotis.



30 pav. Darbo laikas, praleistas įgyvendinant užduotis, naudojant *Kanban*

Sekančiame, 31 paveiksle, pavaizduota, kiek vidutiniškai laiko užtrunkama įgyvendinant vieną istorijos tašką, kiek laiko užtrunkama konkrečios iteracijos laikotarpiu įskaitant tik užbaigtas užduotis ir kiek įskaitant ir neužbaigtas užduotis. Iš grafiko matyti, jog įskaitant tik užbaigtas užduotis yra dvi (5 ir 6 iteracijos) ypač išsiskiriančios vietos ir dar dvi (7 ir 9 iteracijos) šiek tiek mažiau. Visi šie labiau pastebimi išsiskirimai parodo, jog tomis iteracijomis buvo užbaigtos užduotis, kurios buvo pradėtos įgyvendinti dar ankstesnių iteracijų laikotarpiu. Tuo tarpu pateiktas grafikas, kuriame yra įskaitomos ir neužbaigtos užduotys, yra kur kas mažiau išsiskiriantis ir turi net

šešias vietas, kuriose laikas vienam istorijos taškui įgyvendinti praktiškai sutampa su vidutiniu laiko tarpu. Būtent tai parodo *Kanban* metodologijos tolygiškumą ir pastovumą.



31 pav. Darbo laikas, praleistas vienam istorijos taškui įgyvendinti, naudojant *Kanban*

Apžvelgus visus pateiktus grafikus, nesunku pastebėti, jog naudojant *Kanban* metodologiją pritaikius *Scrum* metodologijai būdingą laiko ribojimą, per kurį turi būti atliekamas suplanuotas užduočių kiekis, labai ryškaus skirtumo tarp abiejų metodologijų nesimato ir darbo eiga naudojant *Kanban* taip pat gali atrodyti šiek tiek nepastovi, tačiau pažvelgus į grafikus, būdingus *Kanban* metodologijai, matyti, jog tiek atliekamų darbų, tiek praleisto laiko dirbant grafikai yra tolygesni ir ne tokie „banguoti“ kaip *Scrum*. Iš to galima spręsti, jog naudojant *Kanban* metodologiją pagal jos taisykles darbas būtų atliekamas tolygiau ir nuosekliau.

3.4 Darbo efektyvumo tyrimo analizė

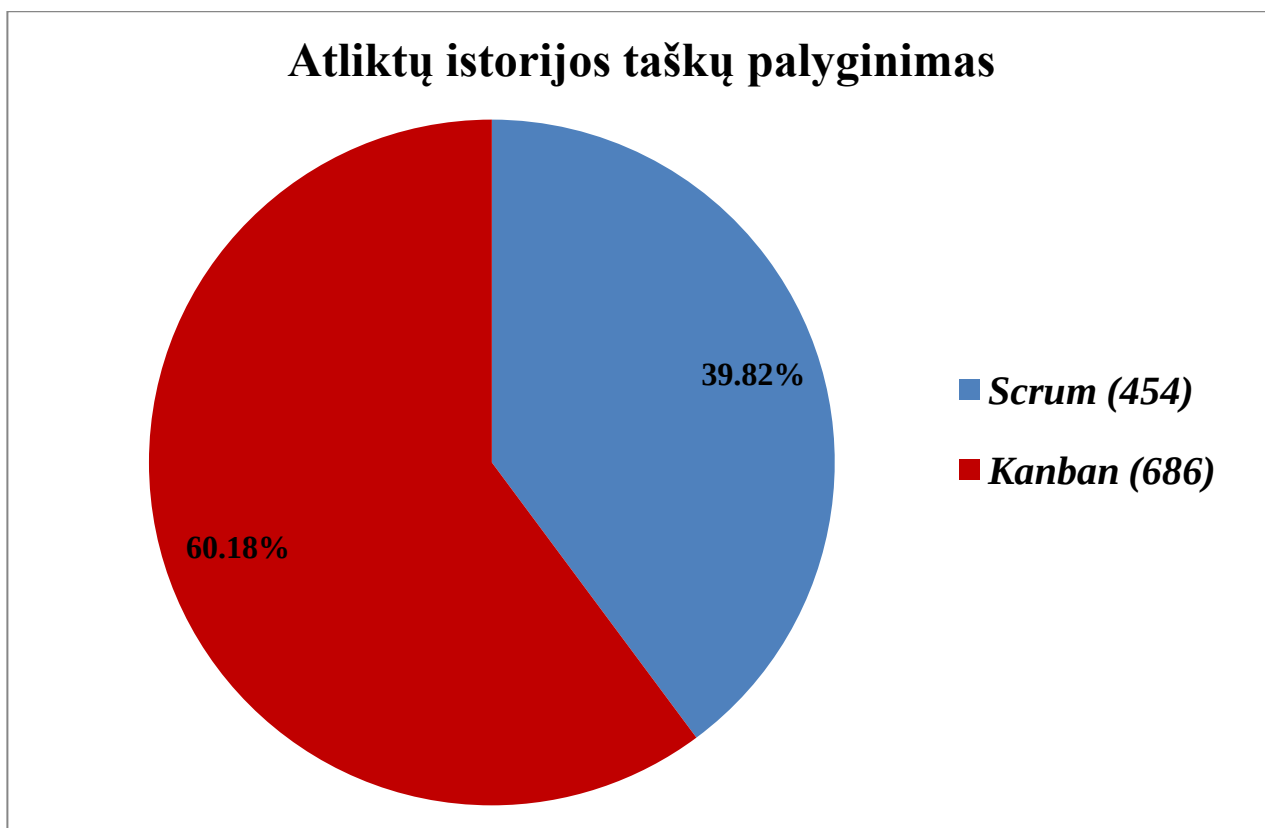
Atlikus *Scrum* ir *Kanban* metodologijų efektyvumo tyrimus, reikia nustatyti, kuri gi iš šių metodologijų yra efektyvesnė. Duomenys, šiai analizei atlikti, buvo paimti tie patys kaip ir atliekant

abiejų metodologijų efektyvumo tyrimus. Efektyvumo tyrimo analizė bus atliekama lyginant šiuos kriterijus:

- Kiek istorijos taškų atlikta per dešimt iteracijų naudojant *Scrum* metodologiją ir kiek naudojant *Kanban*.
- Kiek vidutiniškai istorijos taškų atlikta per vieną iteraciją naudojant *Scrum* metodologiją ir kiek naudojant *Kanban*.
- Kiek laiko praleista dirbant naudojant *Scrum* metodologiją ir kiek naudojant *Kanban*.
- Kiek vidutiniškai užtrunkama laiko atlikti vienam istorijos taškui naudojant *Scrum* metodologiją ir kiek naudojant *Kanban*.

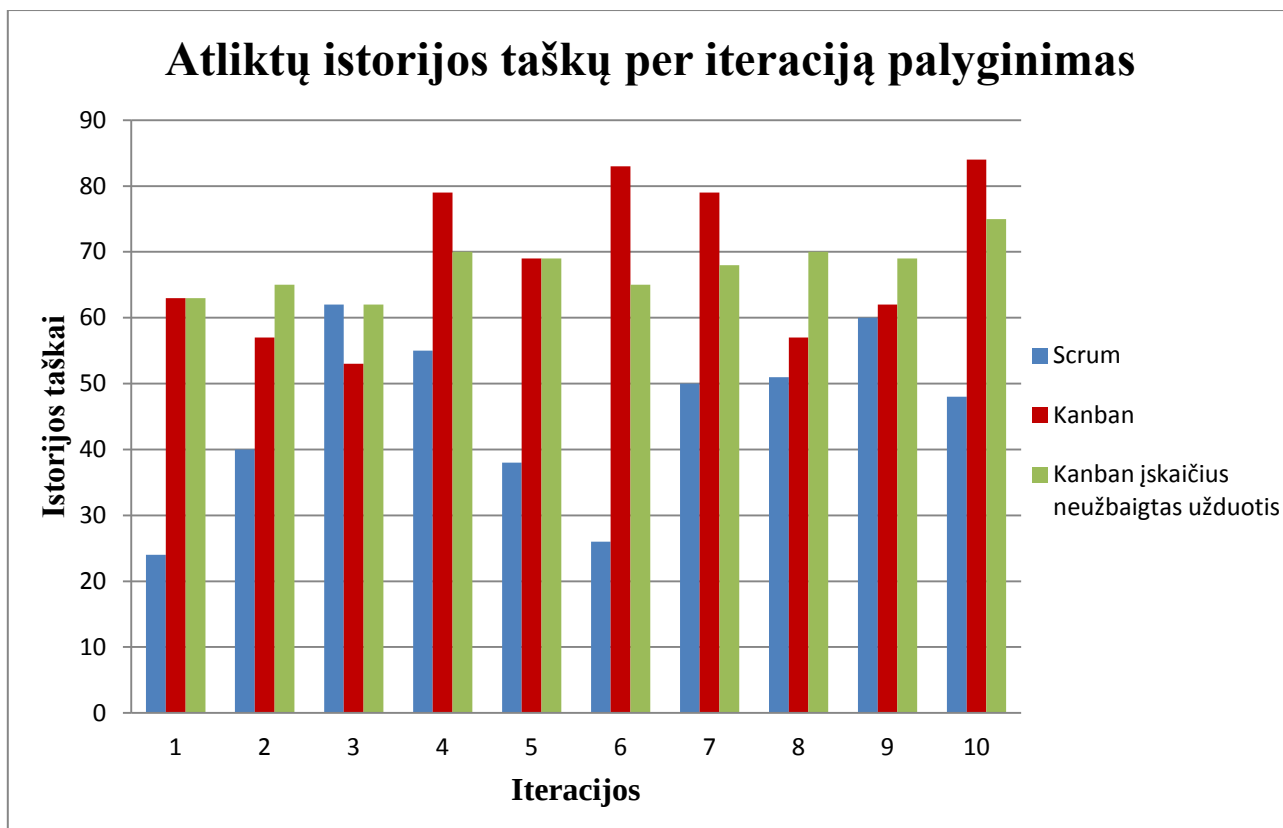
Atlikus palyginimus pagal visus šiuos kriterijus bus galima nustatyti, kuri iš šių dviejų metodologijų yra efektyvesnė ir kurią iš jų taikant galima tikėtis geresnių darbo rezultatų.

Pirmiausia bus palyginama, kurią metodologiją naudojant buvo atlikta daugiau istorijos taškų (32 pav.).



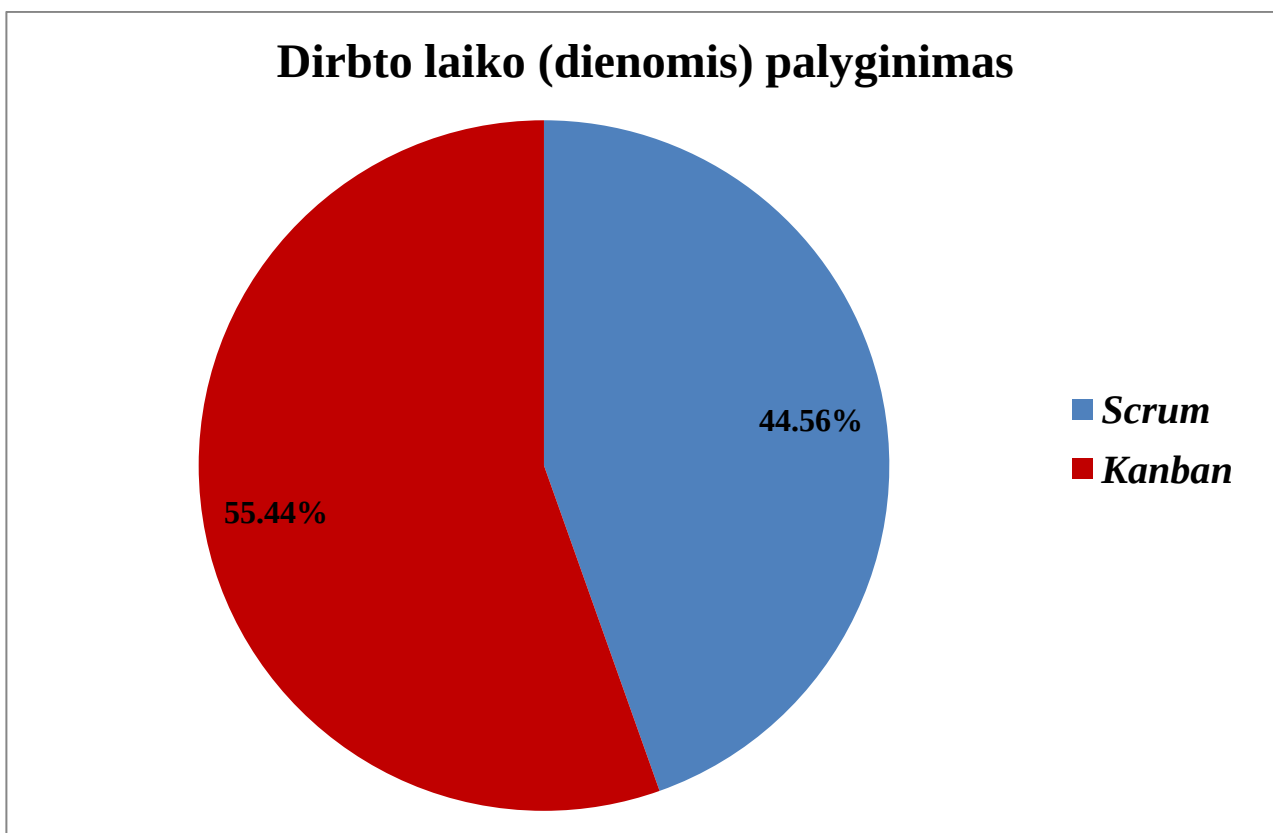
32 pav. Atliktų istorijos taškų palyginimas naudojant abi metodologijas

Palyginus atliktą istorijos taškų skaičių per dešimties iteracijų laikotarpį, tiek naudojant *Scrum* metodologiją, tiek naudojant *Kanban* metodologiją, nustatyta, jog naudojant *Kanban* metodologiją istorijos taškų buvo atlikta 33,81% procentų daugiau, nei per tą patį laikotarpį naudojant *Scrum* metodologiją. Kadangi analizuojamas laikotarpis yra po 7 mėnesius ir 2 savaites kiekvienai metodologijai, o tai nėra labai ilgas laiko tarpas, tai labai tikėtina, kad, ateityje skirtumas tarp atliktų istorijos taškų kiekio naudojant *Scrum* metodologiją ir naudojant *Kanban* metodologiją, tik didėtų. Tokias prielaidas galima daryti pažvelgus į 33 paveikslą, kuriame matyti, jog suskirsčius atliktus istorijos taškus į iteracijas pagal galutinai atliktas užduotis yra vos viena iteracija, kai buvo naudojama *Scrum* metodologija (3), kuri lenkia iteraciją, kai buvo naudojama *Kanban* metodologija, visose kitose iteracijose *Kanban* metodologija yra pranašesnė pagal atliktus istorijos taškus. Maža to, pažvelgus į atliktus istorijos taškus, kai naudojant *Kanban* kiekvienai iteracijai buvo priskiriami ir neužbaigtų užduočių atlikti istorijos taškai, nėra apskritai nė vienos iteracijos, kur *Scrum* metodologija būtų pranašesnė už *Kanban*.



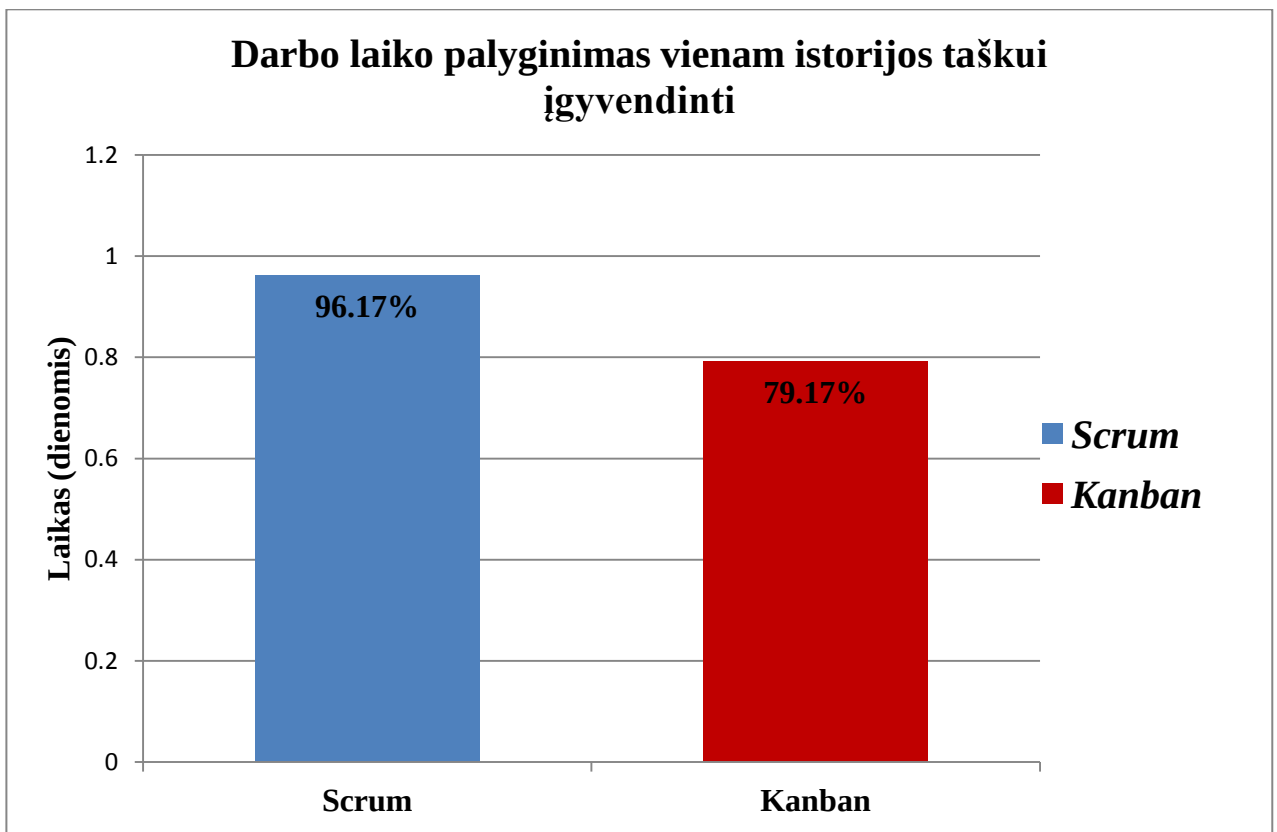
33 pav. Atliktų istorijos taškų per iteraciją palyginimas naudojant abi metodologijas

Verta būtų atkreipti dėmesį ir į praleistą darbo laiką per visą laikotarpį, stengiantis įgyvendinti užduotis, naudojant tiek *Scrum*, tiek *Kanban* metodologijas (34 pav.).



34 pav. Darbo laiko per visą laikotarpį palyginimas naudojant abi metodologijas

Palyginus atliktus istorijos taškus buvo nustatyta, jog naudojant *Kanban*, jų buvo atlikta net 33,81% daugiau, taigi, atrodytų, jog ir darbo laiko turėtų būti praleista daugiau. Taip, naudojant *Kanban* metodologiją, darbo laiko buvo praleista daugiau, tačiau ne 30% daugiau, o viso labo 19,62%. Pagal visus šiuos rodiklius būtų galima spręsti, kad naudojant *Kanban* metodologiją užduočių yra atliekama ne tik daugiau, bet jos atliekamos ir greičiau (35 pav.).



35 pav. Darbo laiko palyginimas vienam istorijos taškui įgyvendinti

35 paveiksle pateiktame grafike matyti, kad darbo laikas, kuris yra sugaišamas vienam istorijos taškui įgyvendinti naudojant *Scrum* metodologiją yra beveik viena diena, tuo tarpu, naudojant *Kanban* metodologiją šis laiko tarpas yra 17,67% trumpesnis negu naudojant *Scrum* metodologiją. Taigi, kuo daugiau istorijos taškų įgyvendinama naudojant *Kanban* metodologiją, tuo daugiau laiko sutaupoma.

3.5 Išvados

Atlikus *Scrum* ir *Kanban* metodologijų efektyvumo tyrimus bei tyrimų analizę nustatyta, jog *Kanban* metodologija yra efektyvesnė už *Scrum* metodologiją. *Kanban* metodologija praktiškai visose tyrimų analizės vietose įrodė savo pranašumą prieš *Scrum* metodologiją, todėl galima teigti, jog norint sutaupyti laiko įgyvendinant užduotis, norint jas atlikti efektyviai, nuosekliai ir kokybiškai, vertėtų rinktis *Kanban* metodologiją.

Išvados ir rekomendacijos

Šiomis dienomis programinės įrangos kūrimas turbūt neįsivaizduojamas be darbo metodologijų naudojimo kūrimo procese. Didžioji dalis programinės įrangos kūrėjų jau suprato darbo metodologijų teikiamą naudą, tačiau nemaža dalis tai supratusių vis dar ieško metodologijų, kurių dėka darbo efektyvumas, našumas, nuoseklumas ir kokybė būtų itin aukšti, o visam tam pasiekti nereiktų skirti daug laiko.

Šiame darbe buvo išanalizuotos dvi gan gerai žinomos metodologijos – tai *Kanban* ir *Scrum*. Atliktoje metodologijų analizėje išsamiai išanalizuotas abiejų metodologijų taikymas, taikymo principai bei ypatybės. Nustatyta, kuo kiekviena metodologija pasižymi ir kokie yra pagrindiniai kriterijai norint pradėti naudoti vieną ar kitą metodologiją. Taip pat nustatyti ir kiekvienos metodologijos privalumai bei trūkumai, su kuriais susiduriama pradėjus naudoti šias metodologijas.

Taip pat buvo atliktas ir abiejų metodologijų efektyvumo tyrimas, kurio metu buvo siekiama nustatyti vienos ir kitos metodologijos efektyvumą. Analizuojant duomenis buvo nustatyta, kiek darbo yra atliekama naudojant vieną metodologiją ir kiek darbo per tokį patį laiko tarpą yra atliekama naudojant kitą metodologiją. Nustatyta, kiek konkrečiai laiko prireikė atlikti visiems darbams naudojant tiek *Kanban*, tiek *Scrum* metodologijas.

Atlikus efektyvumo tyrimą visi duomenys buvo analizuojami dar kartą ir lyginami tarpusavyje. Palyginus atliktų darbų kiekį, buvo gautas rezultatas, jog naudojantis *Kanban* metodologija darbo yra atliekama 33,81% daugiau, negu naudojant *Scrum* metodologiją. Palyginus praleistą darbo laiką įgyvendinant užduotis, gautas rezultatas buvo, jog naudojant *Kanban* metodologiją darbo laiko sugaišta buvo tik 19,62% daugiau, nors darbų atlikta daugiau, o galiausiai palyginus kiek greičiau darbai yra atliekami naudojant *Kanban* metodologiją, gautas rezultatas buvo 17,67%.

Įvertinus visus šiuos gautus rezultatus, atsižvelgus, jog duomenys, kuriais yra paremtas efektyvumo tyrimas, yra paimti iš realaus projekto, galima teigti, jog lyginant šias dvi metodologijas, *Kanban* metodologija yra pranašesnė už *Scrum*, jei norima siekti efektyvaus darbo rezultatų. *Kanban* turi trūkumų, tokių, kaip susirinkimų stoka, tačiau naudojant *Kanban* metodologiją kaip pagrindą ir pridėdant vieną kitą elementą iš *Scrum* metodologijos, darbo rezultatai tikrai turėtų džiuginti tiek darbuotojus, tiek vadotus, tiek, žinoma, klientus.

Literatūra

1. Kniberg, H.; Skarin, M. Kanban and Scrum - making the most of both. 2009.
2. Cohn, M. Succeeding with Agile: Software Development Using Scrum. 2009.
3. Lacey, M. The Scrum Field Guide: Practical Advice for Your First Year (Agile Software Development Series). 2012.
4. Sims, C.; Johnson, H. L. Scrum: a Breathtakingly Brief and Agile Introduction. 2012.
5. Ladas, C. Scrumban - Essays on Kanban Systems for Lean Software Development. 2009.
6. Anderson, D. J.; Reinersten, D. G. Kanban: Successful Evolutionary Change for Your Technology Business. 2010.
7. Klipp, P. Getting Started with Kanban. 2011.
8. Benson, J.; Barry, T. D. Personal Kanban: Mapping Work | Navigating Life. 2011.
9. Why should we choose SCRUM for our projects. [žiūrėta 2012 11 20]. Prieiga per internetą: <http://sandervanpeer.wordpress.com/category/scrum/>
10. Sprint Burndown (Scrum). [žiūrėta 2012 11 23]. Prieiga per internetą: <http://msdn.microsoft.com/en-us/library/vstudio/ff731588.aspx>
11. Release Burndown (Scrum). [žiūrėta 2012 11 23]. Prieiga per internetą: <http://msdn.microsoft.com/en-us/library/vstudio/ff731579.aspx>
12. Advice on Conducting the Scrum of Scrums Meeting. [žiūrėta 2012 11 27]. Prieiga per internetą: <http://www.scrumalliance.org/articles/46-advice-on-conducting-the-scrum-of-scrums-meeting>
13. The Advantages and Disadvantages of Agile SCRUM Software Development. [žiūrėta 2012 12 17]. Prieiga per internetą: <http://www.my-project-management-expert.com/the-advantages-and-disadvantages-of-agile-scrum-software-development.html>
14. Advantages and Disadvantages of the Scrum Project Management Methodology. [žiūrėta 2012 12 18]. Prieiga per internetą: <http://smallbusiness.chron.com/advantages-disadvantages-scrum-project-management-methodology-36099.html>
15. Introduction to Scrum: Benefits and Practices. [žiūrėta 2012 01 03]. Prieiga per internetą: <http://www.cprime.com/knowledge/articles/scrubenefits.html>
16. Successful Scrum implementations. [žiūrėta 2012 01 13]. Prieiga per internetą: http://www.scrumalliance.org/pages/benefits_of_scrum
17. Colin Bird & Rachel Davies Scrum Gathering London 2007. [žiūrėta 2012 01 26]. Prieiga per internetą: www.scrumalliance.org/resource_download/287

18. Kanban. [žiūrēta 2013 01 20]. Prieiga per internetą:
<http://www.crisp.se/gratis-material-och-guider/kanban>
19. Kanban and Scrum. [žiūrēta 2013 01 20]. Prieiga per internetą:
<http://blog.crisp.se/tag/kanban>
20. What is Kanban. [žiūrēta 2013 01 25]. Prieiga per internetą:
<http://www.kanbanblog.com/explained/>
21. Kanban introduction. [žiūrēta 2013 01 25]. Prieiga per internetą:
<http://functionalguy.blogspot.com/2007/04/kanban-introduction-and-benefits.html>

Vilniaus Gedimino Technikos universiteto
Informacinių technologijų katedrai

Magistro darbo praktinis panaudojimas įmonėje
2013 05 22 Nr.13-KOR-S/0028
Vilnius

Baigiamasis magistro darbas "Darbo metodologijų palyginimas bei efektyvumo tyrimas", kurį atliko Evaldas Rekašius, buvo peržiūrėtas ir į jį bus atsižvelgta gerinant procesų ir darbo kokybę įmonėje.

Direktorius



Tomas Vaičiūnas