VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
MATEMATIKOS IR INFORMATIKOS METODIKOS KATEDRA

Jelena Zubova

LAIKO ĮVERČIŲ ANALIZĖ IR VIZUALIZAVIMAS TIRIAMĄJĮ DARBĄ ORGANIZUOJANT REMIANTIS „AGILE“ PRAKTIKOMIS

Magistro baigiamasis darbas

Vadovas
Dr. Rimgaudas Laucius

Leidžiu ginti: _____
(Vadovo parašas)

VILNIUS 2012

TURINYS

ĮVADAS

1. Mokslo problemos aktualumas

Darbo organizavimas ir projektų valdymas yra labai svarbus žmonių kasdienybėje. Neįmanoma gerai atlikti darbo iš anksto neapgalvojus viso proceso, neįvardijus siekiamų tikslų ir priemonių šiems tikslams pasiekti.

Daugelis iš mūsų naudoja įvairius darbo organizavimo metodus net patys to nežinodami. Vieni naudoja dar nuo mokyklos laikų žinomą klasikinį metodą, kiti – kiek pakeistus, pagal savo poreikius pritaikytus metodus.

Darbe pateikta darbo proceso organizavimo ir projektų valdymo metodų apžvalga bei konkretus Scrum metodo pritaikymas leis įvertinti šių metodų galimybes bei palyginti juos su žinomais klasikiniais darbo proceso organizavimo metodais. Darbe pateiktas pagerintas laiko įverčių vizualizavimo metodas suteikia galimybę geriau įvertinti ir prognozuoti projekto sėkmę ir išteklių poreikį.

2. Darbo tikslas

Išanalizuoti darbo bei projektų valdymo metodų procesus, sukaupti ir susisteminti darbo proceso organizavimo žinias ir pateikti metodą, kuriuo remiantis būtų galima sudaryti darbo proceso užbaigimo laikotarpio prognozes.

3. Darbo uždaviniai

- Išanalizuoti mokslinę ir metodinę literatūrą darbo organizavimo procesų ir projektų valdymo metodų klausimais.
- Palyginti kelis dažniausiai naudojamus projektų valdymo metodus su klasikiniu metodu.
- Ištirti darbo organizavimo procesų ir projektų valdymo metodus ir juos susisteminti.
- Sukurti ir aprašyti darbo proceso užbaigimo laikotarpio prognozių sudarymo metodą.
- Pateikti aprašyto darbo proceso vaizdavimo ir užbaigimo laikotarpio prognozių sudarymo metodo praktinį pavyzdį.

4. Naujumas

Darbo organizavimo procesais buvo domėtasi dar XIX a., tačiau per visą šį laiką taip ir nebuvo atkreiptas deramas dėmesys į darbo organizavimo procesų ir projektų valdymo metodų svarbą, nors tai ir yra vienas iš svarbiausių faktorių, įtakojančių darbo ir projektų efektyvų vystymąsi bei visų iškeltų užduočių atlikimą laiku.

Dauguma literatūros šaltinių pateikia darbo organizavimo ir projektų valdymo metodus kaip papildomas priemones, siekiant efektyvesnių bei kokybiškesnių rezultatų. Tačiau apžvelgti metodai turėtų būti ne papildomais, o pagrindiniais darbo organizavimo ir projektų valdymo metodų įrankiais.

Darbe pateiktas pagerintas darbo proceso vaizdavimo metodas leidžia aiškiau perteikti patį procesą ir tiksliau įvertinti jam reikalingus išteklius.

5. Metodai

Darbe naudoti šie mokslinio darbo metodai: 1) informacijos paieška, sisteminimas, analizė ir apibendrinimas, 2) realaus darbo organizavimo proceso aprašymas ir analizė, 3) duomenų analizė ir vizualizavimas, 4) atvejo analizė.

Informacijos paieškos, sisteminimo, analizės ir apibendrinimo metodai naudoti siekiant sukaupti ir pateikti darbo organizavimo bei projektų valdymo metodų žinias.

Realaus darbo organizavimo proceso aprašymas ir analizė naudoti siekiant pateikti realaus proceso eigą bei rezultatus.

Duomenų analizės vizualizavimo metodas naudotas projekto užbaigimo laikotarpio prognozių sudarymo ir vizualizavimo tikslais.

Atvejo analizės metodas naudotas pateikiant konkretų prognozių ir vizualizavimo pavyzdį.

6. Baigiamojo darbo struktūra

Baigiamąjį darbą sudaro įvadas, 8 skyriai, išvados ir literatūros sąrašas.

Įvadas. Tai įvadinis skyrius, kuriame pateikiami darbo tikslai ir uždaviniai, darbo aktualumas ir naujumas, darbe naudoti metodai.

Pirmame skyriuje nagrinėjama darbo organizavimo vystymosi istorija, darbo procesų svarba bei jų reikšmė.

Antrame skyriuje apžvelgiama Agile sąvoka, pateikiami pagrindiniai principai, manifestas bei Agile metodo palyginimas su klasikiniais metodais.

Trečias ir ketvirtas skyriai skirti Agile metodams. Pateiktas išsamus Scrum, Ekstremalaus Programavimo, Kanban bei Lean metodų aprašymas. Šie metodai palyginami su klasikiniu Krioklio metodu.

Penktame, šeštame ir septintame darbo skyriuose pateikiamos realaus darbo proceso Scrum bei Kanban schemos, detalus jų aprašymas, tvarkaraščio sudarymo principai, vykusių susitikimų eigą bei konkretus komandos greičio nustatymo pavyzdys.

Aštuntame skyriuje aprašomi komandos laiko planavimo metodai, proceso diagramų tipai, projekto užbaigimo laikotarpio sudarymo metodas, jo vizualizavimas bei konkretus šio metodo pritaikymas.

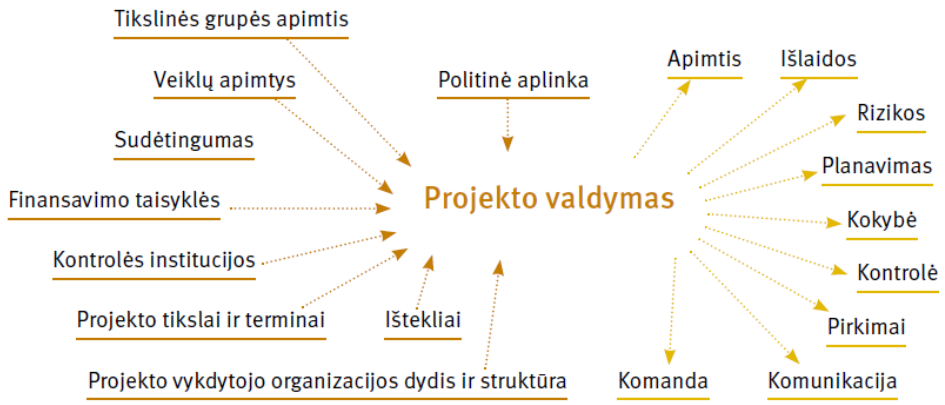
Rezultatai ir išvados. Tai baigiamasis skyrius, kuriame pateikiama darbo apžvalga, bendrosios išvados bei rekomendacijos.

1. DARBO ORGANIZAVIMO PROCESAI, JŲ REIKŠMĖ. PROJEKTŲ VALDYMAS

Darbo organizavimas apibūdinamas kaip priemonių sistema, pagal kurią galima racionaliai panaudoti darbuotojus valdymo procese, taikant racionalų darbo pasidalijimą ir kooperavimą, darbo metodus, darbo vietų organizavimą ir aptarnavimą, darbo sąlygas, darbo normavimą ir materialinį skatinimą [SŠ04].

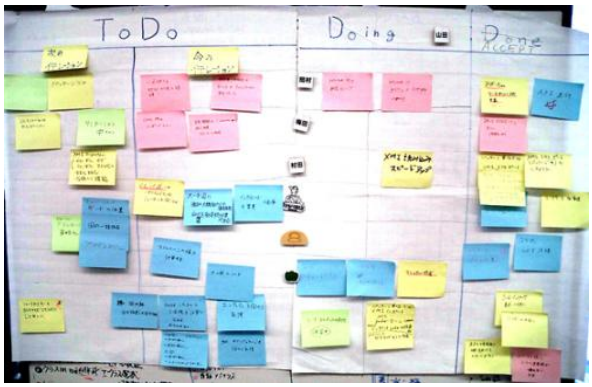
Projektas – laiko ir išteklių ribojama laikina veikla, skirta numatytam tikslui pasiekti. Projektas yra populiarus darbų atlikimo būdas, juo galima apibūdinti įvairius dalykus – nuo paprastų iki pačių sudėtingiausių. Iš esmės tai yra tiksliai laike apibrėžtos, iš anksto suplanuotos ir nuosekliai kontroliuojamos darbo užduotys, kurių pabaigoje pasiekiamas tiksliai pamatuojamas ir įvertinamas rezultatas.

Projekto valdymas – tai procesas, darantis poveikį projekto planavimui, išlaidoms, rizikai, komandos formavimui, pirkimams, apimčiai, kokybei, komunikacijai, kontrolei ir pan. Kiekvieno projekto valdymo procesas yra unikalus, nes kiekvienam projektui daro įtaką skirtingi vidiniai ir išoriniai veiksniai [CPVA10]:



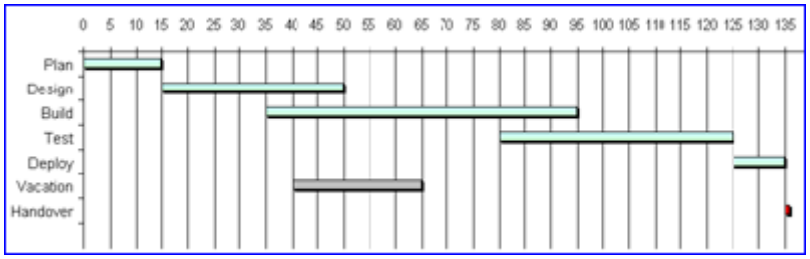
Projekto planas – tai dokumentas, kuriame pateikti projekto tikslai, planuojami veiksmai, jų atlikimo terminai ir kita esminė informacija, kuri iš anksto leidžia visiems suprasti kas ir kaip turi būti padaryta. Skiriant daugiau dėmesio planavimui palengvinamas darbas atlikimo etape. [Pa12]

Labai svarbu pradžioje teisingai suformuluoti projekto tikslą, išskirti projekto užduotis bei suformuluoti jas į darnią sistemą. Tada reikia nustatyti loginę darbų seką ir jų tarpusavio ryšius, t.y. sudaryti schemą. Labai patogi priemonė šiam darbui atlikti yra vizualus pavaizdavimas naudojant Kanban lentą:



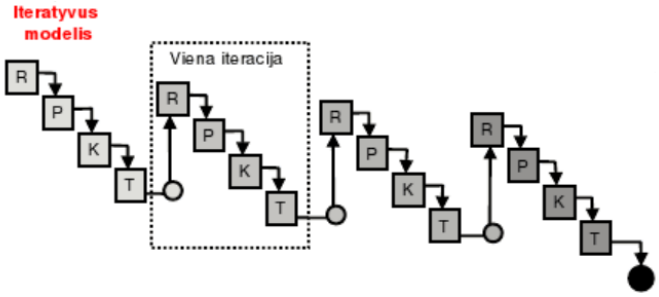
Kiekvienas darbas iš parengto pagrindinių projekto darbų sąrašo užrašomas ant atskiro lapelio. Nusprendžiama, kuris darbas turi būti atliktas pirmiausia - lapelis priklijuojamas ant lentos. Išsiaiškinama, koks darbas arba darbai turi būti atlikti iškart po pirmojo, kurie darbai yra santykinai nepriklausomi ir gali būti atliekami vienu metu, o kurių negalima pradėti, kol nebaigtos ankstesnės užduotys. Tai tęsiama tol, kol visi numatyti darbai neatsiduria bendroje schemoje. [Pa12] Kanban lentoje stulpeliai yra proceso žingsniai, o užduotys dedamos ant lentos tam, kad būtų vizualiai matomos. Kuo greičiau užduotys „perbėgs“ per visus proceso žingsnius, tuo greičiau pasieksime rezultatą.

Išdėsčius darbus logine seka, galima imtis išsamaus darbų tvarkaraščio sudarymo. Yra daug būdų projekto darbų tvarkaraščio sudarymui, tai priklauso nuo projekto sudėtingumo. Vienas iš populiariausių būdų yra Gantt schemos sudarymas. Labai svarbu, kad turima informacija leistų įvertinti, kiek laiko reikia kiekvienai užduočiai atlikti. Pati schema – tai diagrama, kurios vienoje ašyje atidedamas laikas (dienomis, savaitėmis ir pan.), o kitoje ašyje išvardytos užduotys. Gantt schema parodo kiekvienos užduoties trukmę ir atskirų užduočių santykį laike:



Tačiau joje ne visada galima parodyti, kaip atskiri darbai susiję tarpusavyje, koks yra jų loginis ryšys. [Pa12]

Dar vienas darbų tvarkaraščio sudarymo būdas – iteraciniai procesai (sprintai):



Pagrindiniai iteracinio planavimo privalumai yra pakankamai tikslus planavimas, greitesnis produkto atidavimas, didesnė projekto kontrolė, tačiau yra pakankamai sunku nustatyti viso projekto biudžetą ir trukmę.

Bloga projekto komunikacija gali tapti viena iš pagrindinių priežasčių, kodėl kai kurie projektai gali būti įgyvendinti nesėkmingai. Vidinė komunikacija – tai komunikacija tarp projekte dalyvaujančių asmenų ir institucijų. Geresnė komunikacija būna tada, kai projekto dalyviai žino kaip įmanoma daugiau aktualios informacijos, todėl viso projekto įgyvendinimo metu būtina užtikrinti nenutrūkstamą informacijos sklaidą. Projekto dalyviams reikia sudaryti galimybę lengvai prieiti prie aktualios informacijos apie projektą. Vidinės komunikacijos formos – tiesioginis bendravimas, bendravimas telefonu, informacijos sklaida paštu ar el. paštu, pristatymų rengimas, informacijos publikavimas internete ir pan. Jas reikia pasirinkti atsižvelgus į informacijos pobūdį ir skaidos intensyvumą. Gali būti naudojamos kelios komunikacijos formos vienu metu.

Projekto rizika – tai neapibrėžtumas, susijęs su galimybe atsirasti nenumatytoms situacijoms ir su tuo susijusiomis pasekmėmis.

Projektui gali būti aktualu daug rizikų, tačiau ne visos jos būtinai turi atsirasti, o jei atsirado, nebūtinai turi daryti įtaką projektui. Norint atrinkti svarbiausias ir pavojingiausias rizikas, reikia atlikti tikimybės ir poveikio palyginimą. Atlikus šį palyginimą rizikos bus įvertintos pagal svarbą ir įtaką projektui. [CPVA10]

		Poveikis			
		Kritinis	Didelis	Vidutinis	Mažas
Tikimybė	Labai didelė	XXX	XXX	XX	X
	Didelė	XXX	XX	XX	X
	Vidutinė	XX	XX	XX	X
	Maža	X	X	X	X

Rizikas reikia vizualizuoti, kad jas žinotų visa komanda ir pilnavertiškai jas valdytų. Pats lanksčiausias vizualizavimo būdas – padaryti lentą su rizikomis ir sekti jų gyvavimo ciklą. [Bo11]

Rizkų valdymas – tai ne pasyvus nusiteikimas rizikuoti, o metodų ir priemonių visuma, skirta aktyviai paveikti ateitį, siekiant gauti minimalų nuokrypį nuo suplanuotų rezultatų.

Galima išskirti tokius rizikų valdymo etapus:

- rizikos atpažinimas, identifikavimas: nustatomos ir apibūdinamos rizikos, identifikuojamos tiek vidinės, tiek išorinės rizikos. Vidinė rizika dažnai gali būti kontroliuojama arba nesunkiai paveikiama pasitelkus vidinius išteklius. Tuo tarpu išorinė riziką sunkiau identifikuoti ir valdyti.
- nuostolių skaičiavimas, analizė: kokybiškai ir kiekybiškai įvertinamas galimas atsiradusios rizikos poveikis projektui.
- būdų rizikai sumažinti apskaičiavimas ir kontrolė: sudaromas rizikų valdymo ir kontrolės vykdymo planas, numatomi atsakomieji veiksmai, kurie turėtų sumažinti neigiamą rizikų poveikį.

Susidūrus su rizika galima naudoti vieną iš pagrindinių atsako į rizikas strategijų. Pasirinkta strategija turi atitikti riziką:

- vengimo strategija – rizikos pašalinimo ar plitimo sustabdymo veiksmai. Keičiamas projekto planas.
- perkėlimo strategija – rizikos perkėlimas kam nors kitam.
- minimizavimo strategija – išankstiniai veiksmai, kuriais sumažinama rizikos įtaka projektui.
- priėmimo strategija – nesiimama jokių veiksmų ir priimamos rizikos sukeltos pasekmės. [BPĮ07]

2. AGILE PRINCIPAI, MANIFESTAS. AGILE PALYGINIMAS SU KLASIKINIAIS METODAIS

Agile - programų kūrimo metodologijos, pasiūlytos nepelno organizacijos „Agile Alliance[1]“. Agile nėra visiškai nauja metodologija, tai paprasčiausiai „naujo kvėpavimo“ suteikimas seniai žinomoms metodologijoms. Pagrindinė idėja yra projekto lankstumas atsižvelgiant į pastoviai besikeičiančius reikalavimus bei sąlygas.

2001 metų vasario 17 dieną 17 žymių programinės įrangos metodologų grupė[2] pasirašė Agile manifestą[3], kuriame paskelbė, kad labiau vertina:

- žmones ir bendravimą nei procesus ir įrankius,
- veikiančią programinę įrangą nei išsamią dokumentaciją,
- bendradarbiavimą su užsakovais nei kontrakto pasirašymą,
- reagavimą į pakeitimus nei plano vykdymą.

„T. y. nors ir elementai dešinėje pusėje yra vertingi, bet mes vertiname labiau elementus kairėje“[4] [AG01].

Manifestą papildo dvylika principų, kurie paaiškina, ką reiškia Agile:

- mūsų aukščiausias prioritetas yra užsakovo poreikių patenkinimas ankstyvu ir nuolatiniu vertingos programinės įrangos pristatymu.
- kintantys reikalavimai yra sveikintini, net ir vėlyvose kūrimo stadijose. Agile procesai kinta konkurenciniam užsakovo pranašumo užtikrinimui.
- veikianti programinė įranga turi būti pristatoma dažnai, nuo poros savaičių iki poros mėnesių.
- užsakovai ir kūrėjai turi dirbti kartu kiekvieną dieną viso projekto laikotarpio eigoje.
- kurti projektus reikia su motyvuotais asmenimis. Tam, kad darbas būtų atliktas, suteikite jiems aplinką, atitinkančią jų poreikius, ir pasitikėkite jais.
- pats efektyviausias informacijos perdavimo būdas komandai yra asmeninis bendravimas.

---

[1] Agile Alliance – nepelno siekianti organizacija, įsipareigojusi vystyti Agile principus ir praktiką. Agile Alliance palaiko tuos, kurie tiria ir taiko Agile principus ir praktiką, siekiant, kad programinės įrangos pramonė taptų produktyvesne ir humaniškesne.

[2] Kent Beck, Mike Beedlem Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fower, James Grenning, Jim Highsmith, Andrew Kunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas.

[3] Angl. *Agile Manifesto*

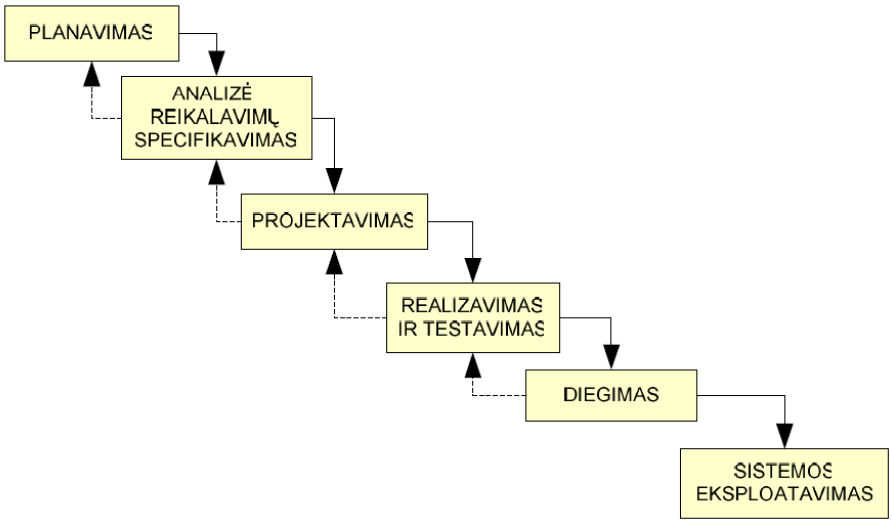
[4] Angl. Citata. *That is, while there is value in the items on the right, we value the items on the left more.*

- veikianti programinė įranga – tai pagrindinis pažangos rodiklis.
- Agile procesai užtikrina stabilią plėtrą. Rėmėjai, kūrėjai ir vartotojai turi visą laiką išlaikyti pastovų tempą.
- pastovus dėmesys technikos tobulinimui ir geram dizainui pagerina Agile procesus.
- svarbus paprastumas – darbo apimties padidinimo išvengimas.
- pačios geriausios architektūros, reikalavimai ir dizainas sukuriami savarankiškai besiorganizuojančiomis komandomis.
- po reguliarių laiko intervalų komanda turi susimąstyti apie tai, kaip tapti efektyvesne ir atitinkamu būdu koreguoti savo elgesį. [AP01]

Egzistuoja gana daug projektų valdymo modelių, labiau paplitę yra šie: Krioklio[5], Fontano[6], Spiralės[7]. Labai dažnai metodikos yra grupuojamos pagal projekto gyvavimo ciklą. Projekto gyvavimo ciklą galima apibrėžti kaip tam tikrų etapų rinkinį, kuris apima visą projekto laikotarpį nuo projekto pradžios iki pat pabaigos. [BK09]

Krioklio modelis yra grindžiamas nuosekliu visų projekto gyvavimo ciklo etapų vykdymu, kiekvienas projekto etapas turi būti užbaigtas prieš pradedant sekantį projekto etapą. Krioklio modelio projektų valdymas orientuotas į projekto tikslų pasiekimą nustatytuose rėmuose.

Projekto pradžioje yra sudaromas detalus planas ir jo vykdymas yra vienas iš projekto kontrolės ir sėkmės vertinimo kriterijų, todėl šiuo metodu valdomi projektai kartais įvardijami kaip projektai, varomi projekto planu. [BK09]



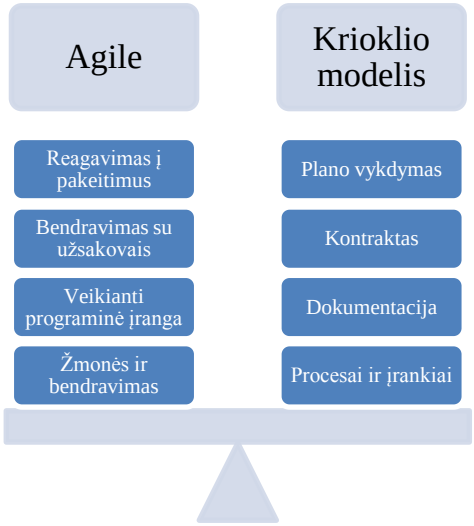
[5] Angl. *Waterfall model*

[6] Angl. *Fountain model*

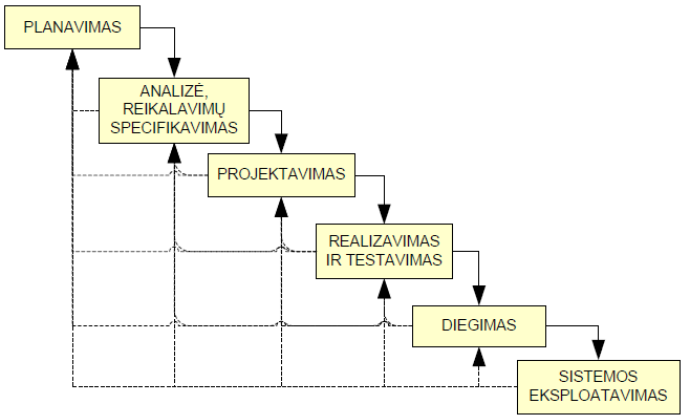
[7] Angl. *Spiral model*

Dažniausiai Krioklio modelis naudojamas sudėtingoms sistemoms kurti, kai reikalavimai sistemai yra žinomi ir apibrėžti dar prieš sistemos kūrimo etapo pradžią. [Tu08]

Agile ir Krioklio modelių palyginimas:



Fontano modelis – tai tam tikra Krioklio modelio modifikacija. Kaip ir Krioklio modelyje čia iš eilės vykdomi visi etapai, tačiau skirtumas tas, kad iš kiekvieno etapo egzistuoja grįžtamasis ryšys į prieš tai buvusius etapus:

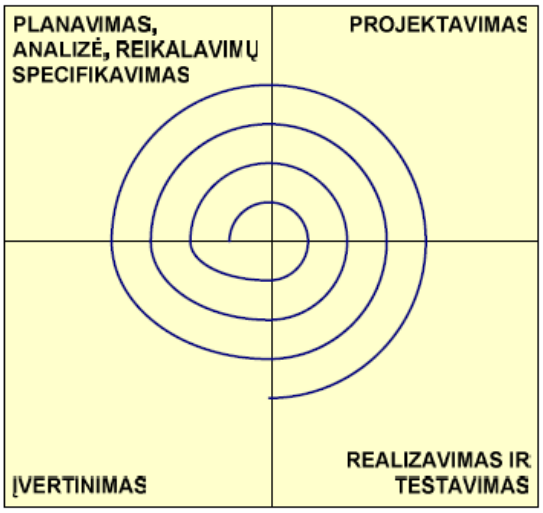


Šis metodas dar kartais vadinamas Iteratyviu krioklio modeliu[8] arba Kaskadiniu modeliu[9].

Spiralės modelio esmė – keturių etapų iteratyvus vykdymas [Tu08]:

[8] Angl. *Iterative Waterfall model*

[9] Angl. *Cascade model*



Spiralės modelis yra tarsi trumpų Krioklio modelio ciklų serija, realizacija vykdoma palaipsniui – iš pradžių realizuojamos pagrindinės funkcijos, sulig kiekviena nauja spirale įdiegiamos vis mažesnės reikšmės funkcijos.

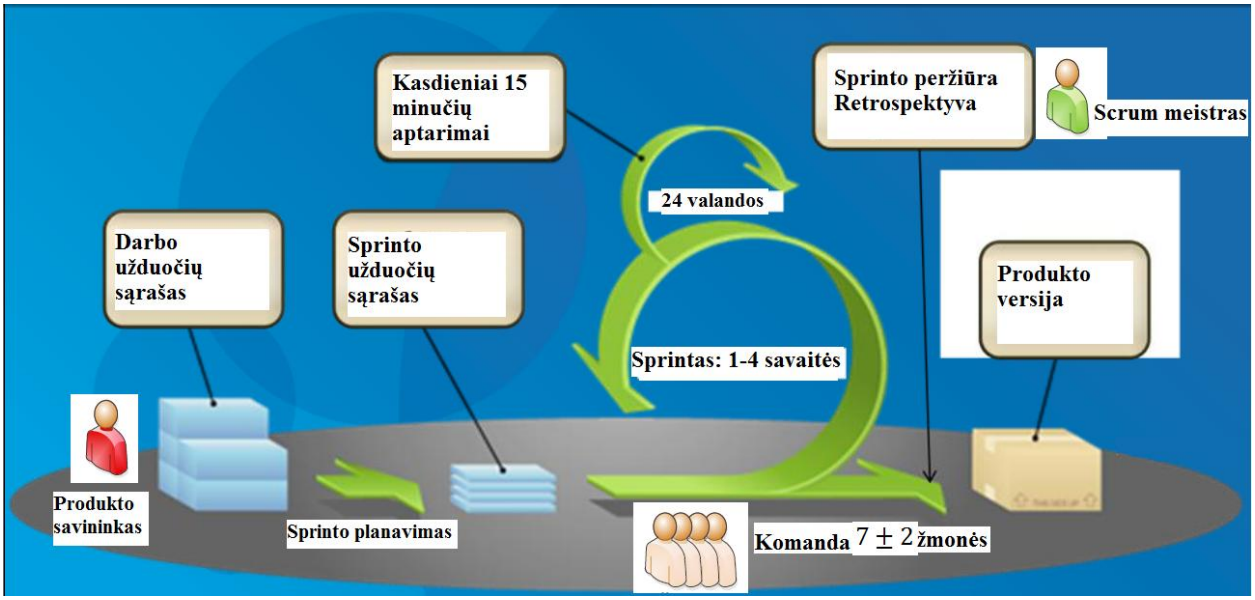
Agile metodų pagalba ilgalaikiai projektai skaidomi į smulkesnius tarpinius projektus, kuriuose dirbtų mažos komandos (2-3 žmonės). Viena projekto iteracija trunka nuo 1 iki 4 savaičių, kiekvienos iteracijos pabaigoje peržiūrimi projekto prioritetai, tai leidžia greitai prisitaikyti prie kintančių reikalavimų bei suvaldyti atsirandančias rizikas.

Pagrindinis Agile ir klasikinių metodų skirtumas yra tas, kad Agile orientuotas į principus, o klasikiniai metodai orientuoti į procesą.

3. AGILE METODAI

3.1. Scrum metodas

Scrum – vienas iš populiariausių Agile metodų. Iš esmės, Scrum reiškia iteracinį saviorganizuojančių ir savarankiškų komandų darbą.



Darbas organizuojamas nedidelėse komandose, kuriose išrenkamas Scrum meistras atsakingas už procesų laikymąsi komandoje. Reikalavimai skaidomi į nedideles dalis, kurios maksimaliai nepriklausytų viena nuo kitos, rezultate gaunamas produkto užduočių sąrašas. Paskui darbo užduočių sąrašo elementai surikiuojami svarbos tvarka. Visas darbas suskaidomas trumpomis (1-4 savaitinėmis) iteracijomis – sprintais, kiekvieno iš jų pabaigoje pristatoma produkto versija. Kiekvieną dieną vyksta aptarimai. Jie skirti tam, kad visi komandos nariai žinotų, kas ir kuo užsiima šiame projekte. Šių aptarimų trukmė griežtai apribota ir neturi viršyti 15 minučių. Kasdienių aptarimų tikslas – informacijos sklaida. Aptarimai nėra skirti problemų sprendimų aptarimui. Kiekvienam komandos nariui yra užduodami trys klausimai:

- Kas buvo padaryta vakar?
- Kas bus padaryta šiandien?
- Su kokiomis problemomis buvo susidurta? [SD12]

Darbo proceso eigoje komandos nariai pasirenka elementus iš sprinto užduočių sąrašo atsižvelgiant į nustatytus prioritetus.

Kiekvieno sprinto pabaigoje organizuojama sprinto peržiūra. Sprinto peržiūros susitikimo metu komanda demonstruoja produktą, sukurtą per paskutinį sprintą. Susitikimo metu daromos išvados, kaip toliau turėtų vystytis sistema bei pateikiami proceso gerinimo pasiūlymai. [SS12]

Klasikinis Scrum metodas susideda iš tokių elementų:

Scrum dažniausiai išskiriami trys vaidmenys:

- Produkto savininkas[10] - žmogus atsakingas už reikalavimų sukūrimą ir jų prioritetų nustatymą.
- Scrum meistras – komandos narys, kuris yra atsakingas už procesus, komandinio darbo koordinavimą.
- Komanda – 7±2 žmonės, realizuojantys produkto savininko reikalavimus.

Artefaktai:

- Darbo užduočių sąrašas[11] - reikalavimų, išdėstytų prioritetų tvarka, sąrašas.
- Sprinto užduočių sąrašas[12] - produkto užduočių sąrašo dalis, atrinkta sprintui.
- Produkto versija – naujas produkto funkcionalumas, sukurtas paskutinio sprinto metu.

Procesai:

- Sprinto planavimas. Jo rezultate gaunamas užduočių, kurias komanda planuoja padaryti sprinto metu, sąrašas.
- Sprinto peržiūra – produkto, sukurto paskutinio sprinto metu, demonstravimas.
- Retrospektyva vyksta po sprinto peržiūros. Surenkama visa komanda sprinto rezultatų aptarimui.
- Kasdieniai susitikimai[13] - komandos narių susirinkimai. [Bo11]

3.2. Ekstremalus programavimas

Ekstremalus programavimas[14] dažnai žymimas santrumpa XP – tai vienas iš Agile metodų programinei įrangai kurti. Šis metodas tinka nedidelėms ir vidutinio dydžio komandoms, kuriančioms programinę įrangą neaiškių arba greitai besikeičiančių reikalavimų sąlygomis.

---

[10] Angl. *Product owner*

[11] Angl. *Product Backlog*

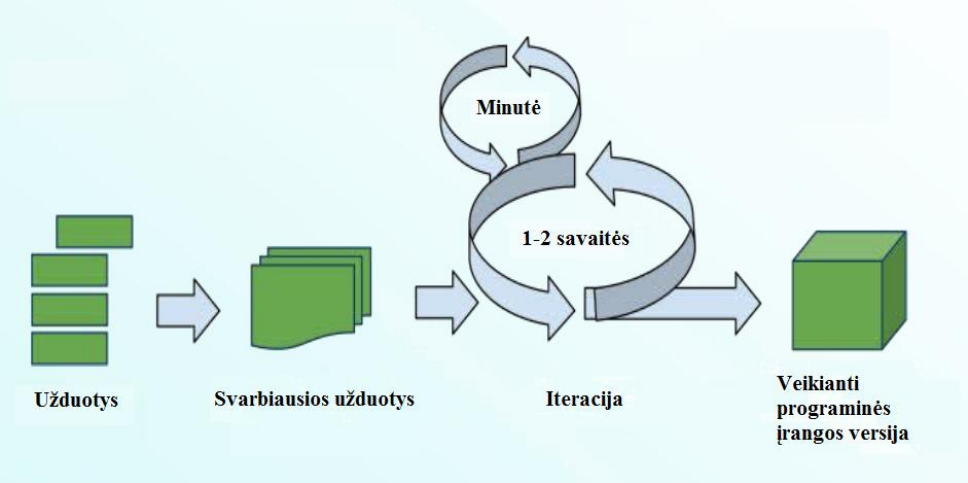
[12] Angl. *Sprint Backlog*

[13] Angl. *Scrum Meeting*

[14] Angl. *Extreme Programming*

Pagrindiniai ekstremalaus programavimo tikslai yra padidinti užsakovo pasitikėjimą programine įranga pateikiant realius sėkmingo proceso vystymosi įrodymus bei staigus produkto kūrimo terminų sutrumpinimas. Tuo pačiu XP sutelktas į klaidų minimizavimą ankstyvose kūrimo stadijose. Tai leidžia pasiekti maksimalų produkto sukūrimo greitį. Praktiškai visos XP praktikos yra sutelktos į programinės įrangos kokybės gerinimą. [In12]

Ekstremalaus programavimo projektas:



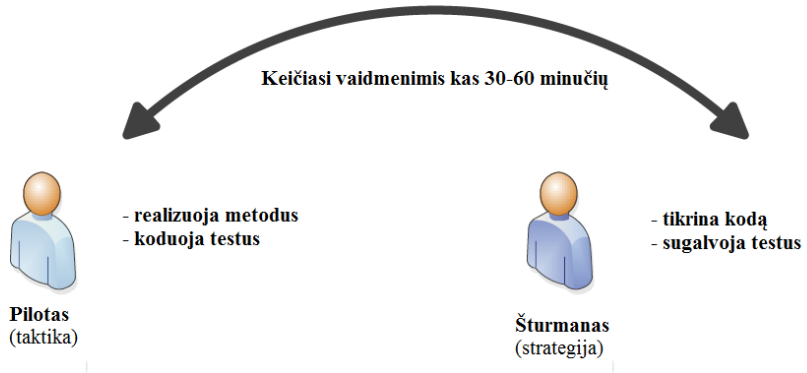
Pagrindiniai ekstremalaus programavimo principai:

- Iteracijos. Kūrimas vyksta trumpomis iteracijoms, rekomenduojama trukmė – 2-3 savaitės ir ne daugiau kaip 1 mėnuo. Vienos iteracijos metu komanda turi kelias sistemos savybes, kiekviena iš jų turi būti aprašyta vartotojo istorijoje. Vartotojo istorija – tai pradinė informacija, kurios pagrindu sudaromas modulis.
- Sprendimų paprastumas. Metodo ekstremalumas susijęs su didele sprendimų priėmimo rizika, sąlygota griežtu laiko grafiku. Pirmoje ir kiekvienoje sekančioje iteracijoje realizuojamas minimalus pagrindinių sistemos funkcijų rinkinys; sulig kiekviena kita iteracija funkcionalumas plečiasi.
- Intensyvus kūrimas mažomis grupėmis (ne daugiau 10 žmonių) ir programavimas poromis (kai du programuotojai kartu kuria kodą dirbdami prie vieno kompiuterio), aktyvus bendravimas grupėje ir tarp grupių. Viso šito siekis yra kaip galima anksčiau aptikti problemas bei klaidas.
- Pastovus grįžtamasis ryšys – bendravimas tarp komandos ir užsakovo.
- Drąsa ir noras rizikuoti.

Ekstremalus programavimas turi 12 taisyklių, kurių privaloma laikytis norint pasiekti geriausią rezultatą:

1. Proceso planavimas. Visa kūrėjų komanda susirenka kartu ir priima sprendimą, kokios sistemos savybės bus realizuotos per artimiausią iteraciją.

2. Nuolatinis grįžtamasis ryšys. Užsakovas turi būti XP komandos nariu. Jis rašo vartotojo istorijas, išrenka tas istorijas, kurios bus realizuotos konkrečioje iteracijoje.
3. Bendras supratimas. Kūrėjų komanda turi turėti bendras taisykles ir jomis vadovautis.
4. Paprasta architektūra. Bet kuri sistemos savybė turi būti realizuota kuo paprasčiau. XP komandos programuotojai dirba vadovaudamiesi devizu: „Nieko nereikalingo!“.
5. Pertvarkymas – tai kodo gerinimo metodika, nepakeičiant jo funkcionalumo. [In12]
6. Programavimas poromis – du programuotojai dirba prie vieno kompiuterio: vienas iš programuotojų gauna „piloto“ vaidmenį, kitas – „šturmano“: [Bo11]



7. 40 valandų darbo savaitė. Programuotojas neturi dirbti daugiau kaip 8 valandas per dieną. Viršvalandžiai – ryškus konkretaus kūrimo etapo problemos indikatorius.
8. Bendra kodo nuosavybė. Kiekvienas komandos narys yra atsakingas už pirminį programos kodą. Tokiu būdu kiekvienas gali keisti bet kurią programos dalį. Svarbi taisyklė: jeigu programuotojas kažką pakeitė ir sistema pradėjo veikti nekorektiškai, tai būtent tas programuotojas turi ištaisyti klaidas.
9. Kodavimo standartai. Visi komandos nariai turi laikytis bendrų kodavimo standartų, kurių dėka programos kodas būtų suprantamas kiekvienam nariui.
10. Nedidelės laidos[15]. Kuo dažniau išleidžiamos produkto versijos, tuo daugiau sistemos trūkumų bus aptikta. Pirmosios laidos padeda aptikti trūkumus pačiose ankstyviausiose stadijose.
11. Nuolatinis integravimas. Visos sistemos kodo integravimas atliekamas kelis kartus per dieną, po to, kai kūrėjai įsitikina, kad visi testai sėkmingai veikia. Dažnas sistemos kodo integravimas padeda išvengti rimtesnių problemų. [In12]
12. Testavimas. Didelis dėmesys skiriamas dviem testavimo rūšims:

[15] Angl. *Releases*

- modulių testavimas[16];
- priėmimo testavimas[17];

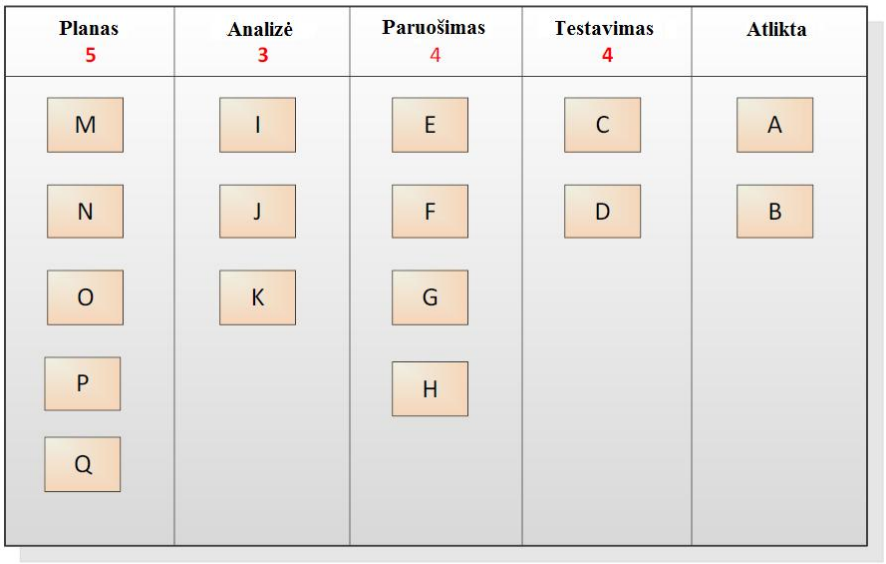
Kūrėjas gali būti tikras, kad kodas parašytas gerai tik tada, kai visi kuriamos sistemos modulių testai veikia gerai. Modulių testai leidžia įsitikinti, kad kodas veikia gerai. Jie taip pat padeda suprasti kitiems programuotojams, kam reikalingas vienas ar kitas kodo fragmentas ir kaip jis funkcionuoja. Priėmimo testai leidžia įsitikinti tuo, kad sistema iš tikrųjų atitinka keliamus reikalavimus bei leidžia patikrinti kuriamo produkto funkcionalumo korektiškumą. [Wi12]

Ekstremalaus programavimo procesas yra neformalus, tačiau reikalauja griežtos disciplinos. Jeigu šios taisyklės nesilaikoma, XP tampa chaotišku ir nekontroliuojamu procesu. Kiekvienas XP komandos programuotojas yra kvalifikuotas darbuotojas, kuris profesionaliai ir su didele atsakomybe atlieka savo pareigas. [In12]

3.3. Kanban metodas

Kanban – tai vienas iš Agile metodų, kuris koncentruojasi į proceso tėkmės optimizaciją. Kanban – tai adaptyvus instrumentas, kuris reikalauja iš komandos, nusprendusios jį naudoti, aukšto lygio saviorganizacijos ir disciplinos. Norint naudoti šį metodą, reikia vadovautis tik trimis taisyklėmis:

- darbo proceso vizualizavimas. Šiuo tikslu dažniausias naudojama Kanban lenta, kurioje stulpeliai yra proceso žingsniai. Visos projekto užduotys dedamos ant lentos, kad būtų vizualiai matomos. [Bo11]



[16] Angl. *Unit Testing*

[17] Angl. *Acceptance Testing*

Plano stulpelis: čia laikomos užduotys, paruoštos atlikimui. Pirma visada imama aukščiausiai lentoje esanti prioritetinės užduoties kortelė ir perkeliama į kitą stulpelį.

Analizės stulpelis: šis ir kiti stulpeliai, esantys iki stulpelio „Atlikta“, gali keistis, kadangi komanda pati sprendžia, kokius žingsnius užduotis turi pereiti iki atlikimo būsenos. Pavyzdžiui, šiame stulpelyje gali būti užduotys, kurių reikalavimus analizuoja ir detaliai išstudijuoja analitikas. Kai analizė baigta, užduotis perkeliama į kitą stulpelį.

Paruošimo stulpelis: čia užduotis lieka tol, kol yra paruošiama.

Testavimo stulpelis: šiame stulpelyje užduotis yra testuojama. Jeigu randamos klaidos – užduotis grąžinama į paruošimo stulpelį, jei ne – perkeliama toliau.

Atlikimo stulpelis: čia užduotis atkeliauja tik tada, kai visi darbai yra pilnai užbaigti. [Xa12]

- proceso optimizavimas. Būtina matuoti užduočių atlikimo (užduoties „prabėgimas“ per visus proceso žingsnius) laiką ir jį mažinti.
- pradėtų užduočių skaičiaus ribojimas. Kiekvienas stulpelis turi turėti po skaičių, kuris nurodytų, kiek daugiausiai užduočių gali būti tame stulpelyje. Tai užtikrina, kad per daug darbo nebus perduota į sistemą ir ji nepradės lėtėti.

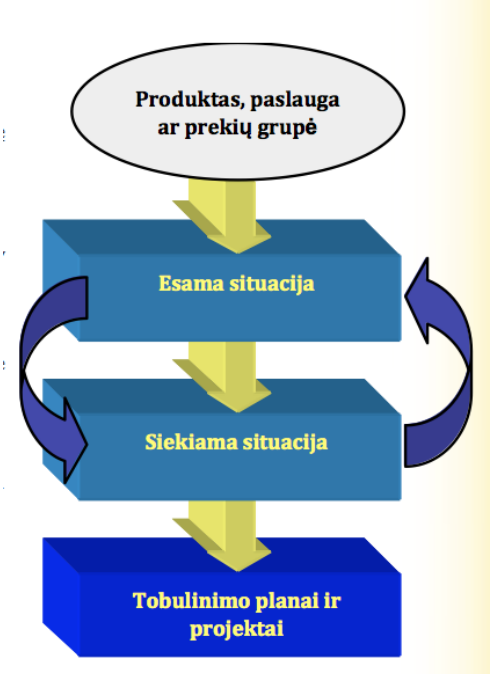
3.4. Lean metodas

Lean – tai vienas iš Agile metodų, padedantis mažiausiomis įmanomomis sąnaudomis pagaminti produktą bei pašalinti pridėtinės vertės nekuriančius proceso žingsnius. Pagrindiniai Lean principai:

- Nuostolių eliminavimas. Nuostoliai – tai bet kokia veikla, kuri naudoja resursus, bet nesukuria vertės. [De12] Galima išskirti 10 nuostolių rūšių:
    - Taisymas/koregavimas
    - Produkcijos perteklius
    - Bereikalingas darbas
    - Bereikalingas daiktų judėjimas
    - Bereikalingas žmonių ir įrangos judėjimas
    - Bereikalingas inventorius
    - Laukimas/prastovos
    - Talentų neišnaudojimas
    - Darbų saugos trūkumas
    - Bet koks remontas [IS11]
- Mokymosi svarba. Visas programinės įrangos kūrimo procesas faktiškai yra tam tikras mokymasis: sulig kiekvieną naują funkciją sužinoma kažkas naujo. Todėl labai svarbu pabrėžti mokymosi svarbą.

- Kokybė. Labai svarbus kokybės užtikrinimas ir jos kontrolė.
- Vėlyvas sprendimų priėmimas. Svarbu sprendimus priimti kaip galima anksčiau.
- Greitas pristatymas. Taupus programinės įrangos kūrimas numato maksimaliai greitus ir dažnus pristatymus, kas ypač svarbu grįžtamojo ryšio palaikymui.
- Pagarba žmonėms. Komanda turi būti gerbiama.
- Sistemos optimizavimas. Turi būti optimizuojama visa sistema, o ne tik atskiros jos dalys. [De12]

Bendra Lean metodo schema:



Pirmas žingsnis – produkto ar produktų grupės pasirinkimas, projekto apimties bei procesų, kuriuos jis apims, nustatymas.

Antras žingsnis – esamos situacijos nustatymas. Informacijos ir duomenų, kurie atspindėtų realią situaciją, užtikrinimas. Remiantis realia situacija bus sudaromas siekiamos situacijos „procesų žemėlapis“. [IS11]

Trečias žingsnis – sudaroma procesų schema – procesų vertės srauto žemėlapis:

- Analizuojama esama situacija. Vykdomas procesas skaidomas į užduotis ar operacijas. Kiekviena operacija užrašoma ant atskiro lapelio. Visi lapeliai tam tikra tvarka priklijuojami ant lentos ar popieriaus lapo – sukuriamas vertės srauto žemėlapis.

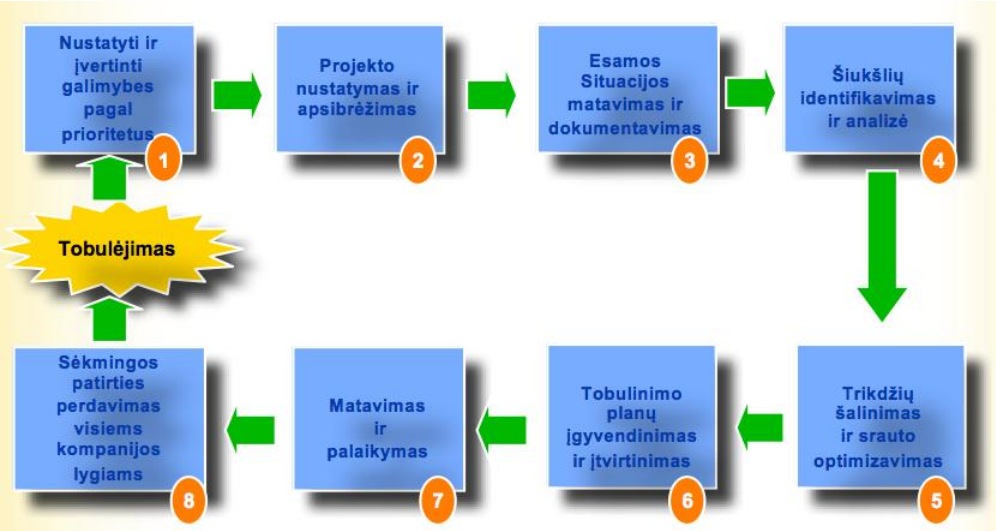
- Modeliuojamas siekiamos situacijos procesas. Analizuojama kiekvienos operacijos „vertė“ ir pašalinamos vertės nekuriančios operacijos – atitinkami lapeliai nuimami nuo žemėlapio. Tokiu būdu sukuriamas optimalaus proceso žemėlapis.

- Sudaromas proceso optimizavimo planas operacijų tvarkos keitimo bei nuostolių šalinimo būdu.

Siekiama situacija kuriama remiantis esama situacija, pagal numatomus tobulinimo projektus ir planus. [Su12]

Paskutinis žingsnis – užtikrinimas, kad būtų pasiektas tobulumo planas ir viziją, kurie realiai optimizuotų kaštus, ciklo laiką ir kokybę. [IS12]

Lean projektai įgyvendinami pagal tokią schemą:



1 etapas: nustatyti ir įvertinti galimybes pagal svarbą. Pirmiausia išrenkamos problemos, kurių sprendimas gali atnešti daugiau naudos ir įgyvendinimas nereikalauja kardinalių pokyčių.

2 etapas: projekto pasirinkimas ir apimčių nustatymas. Sudaroma Lean projekto komanda, apibrėžiama projekto vieta, apimtys ir pan.

3 etapas: esamos situacijos matavimas ir dokumentavimas. Pagal esamą situaciją sudaroma detali procesų schema. Į šią schemą įtraukiami aktualūs duomenys ir kiti parametrai. Esama situacija užfiksuojama.

4 etapas: proceso šiukšlių identifikavimas ir šalinimas. Tai analizės etapas, kurio tikslas rasti proceso šiukšles, jų atsiradimo priežastis bei numatyti būdus joms šalinti.

5 etapas: trikdžių šalinimas ir srauto optimizavimas. Naikinamos proceso šiukšlių atsiradimo priežastys bei vizualizuojamas darbas ir procesai. Tokiu būdu pasiruošiama tobulinimo procesams, t.y. išgryninamas darbas ir procesai.

6 etapas: tobulinimo planų įgyvendinimas ir įtvirtinimas. Tai realus ir fizinis tobulinimo planų įgyvendinimas.

7 etapas: matavimas ir palaikymas. Pagal naują esamą situacija sudaroma detali procesų schema. Įtraukiami aktualūs duomenys ir parametrai. Užfiksuojama esama situacija ir lyginama, koks pagerėjimo lygis pasiektas.

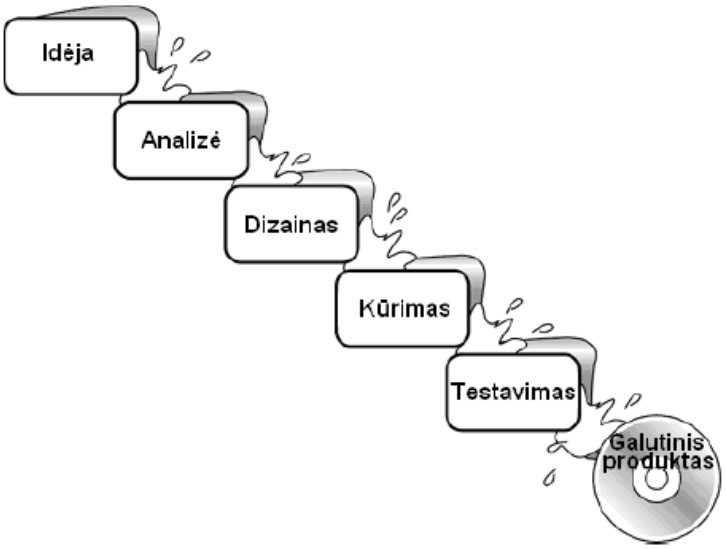
8 etapas: gerosios praktikos sklaida. Gerosios praktikos sklaida visada svarbi, o Lean projektų atveju svarbu, kad visi kompanijos darbuotojai vystytų gebėjimus palaikyti pasiekto pagerėjimo lygį. [ID12]

Taigi, Lean – tai požiūris, padedantis matyti ir spręsti problemas projekto kūrimo metu, bei nuolatinio tobulėjimo filosofija.

4. AGILE METODŲ PALYGINIMAS SU KLASIKINIU KRIOKLIO METODU

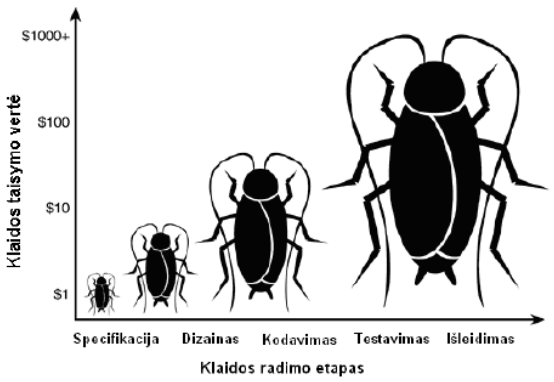
Krioklio modelis laikomas klasikiniu modeliu, kuriame nuosekliai išskiriamos pagrindinės veiklos grupės, skirtingų komandų atliekami etapai.

Krioklio modelis yra paprastas, nuoseklus bei formalizuotas. Kūrimo procesas vyksta aiškiai ir disciplinuotai. Didelis dėmesys skiriamas reikalavimų analizei, projekto dokumentacijai. Kuriant Krioklio principu, kiekviena nauja stadija pradedama, kai baigiama ankstesnė.



Kadangi projekto rezultatas matomas tik pabaigoje, reikalaujama kvalifikuotos komandos ir labai aiškių bei visiems suprantamų reikalavimų. Projekto testavimas atliekamas tik pabaigoje, todėl klaidos brangiai kainuoja, nes kuo vėlesniame etape klaida atrandama, tuo jos vertė projekte didesnė – taisymas užtrunka ilgiau, įtraukiama daugiau žmonių ir pan.

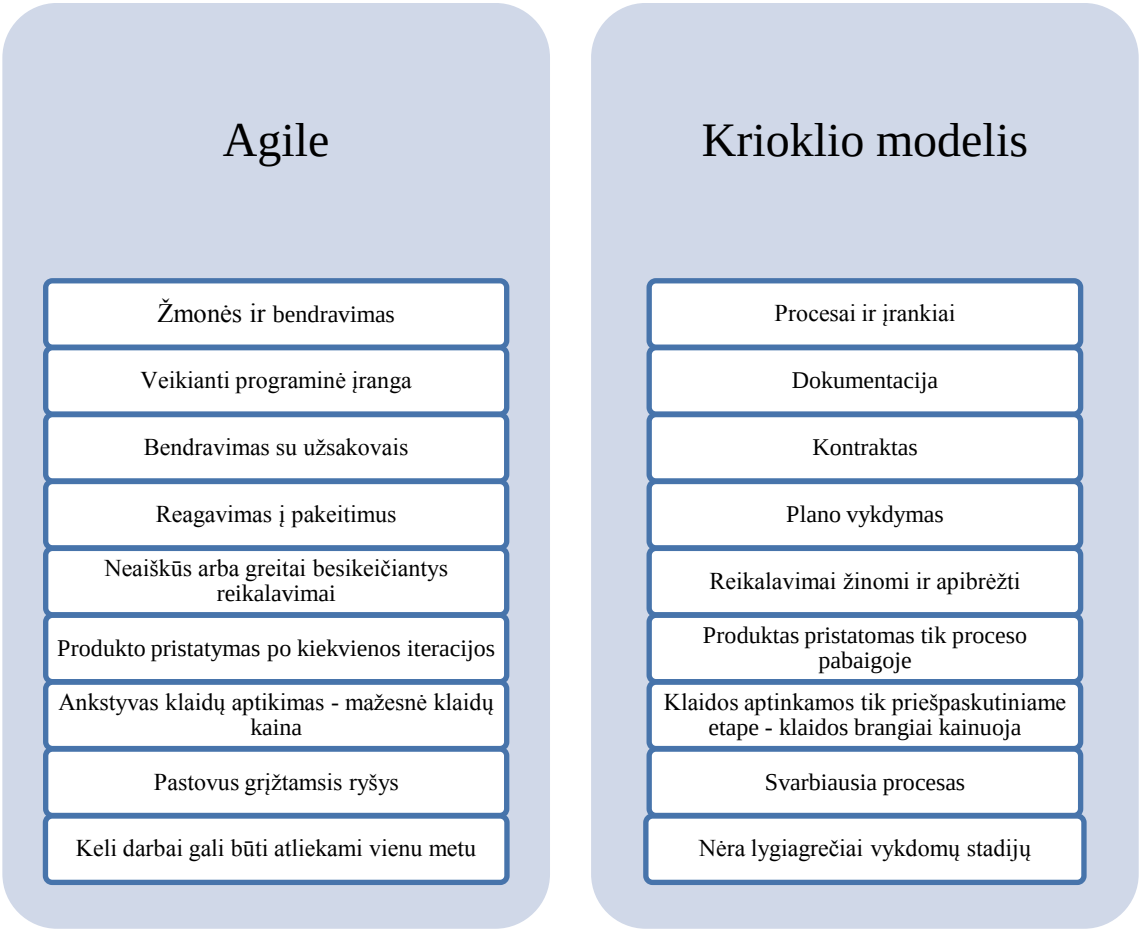
Kai rezultatą matome tik pabaigoje ir testavimas vyksta paskutiniame etape – klaidų kaina yra didelė.



Modeliui įgyvendinti turi būti iš anksto puikiai žinomi nesikeičiantys reikalavimai, kas netinka dideliems projektams, kur svarbu įvertinti ir valdyti besikeičiančius reikalavimus ir mastus.

Agile metodų tikslas sutelktas į lankstumą. Agile metodai sugeba prisitaikyti prie aplinkybių, leidžia kuo lengviau vykdyti pokyčius ir juos valdyti. Agile metodų pagalba produktas kuriamas pagal viziją, o ne iš anksto sudarytą ilgalaikį planą. [Šk09]

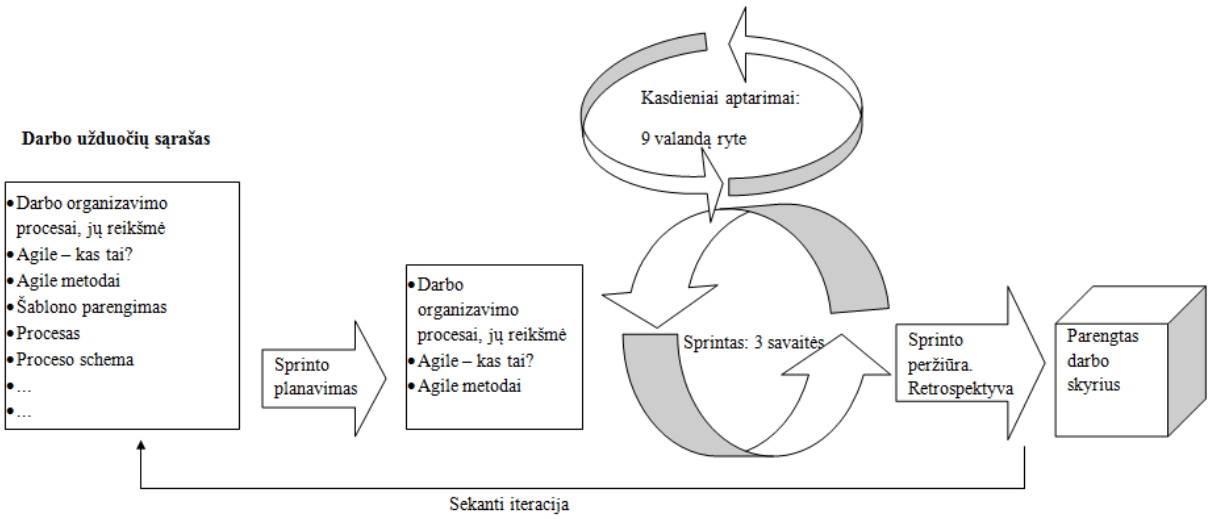
Agile ir Krioklio modelių palyginimas:



Krioklio modelyje labai sunku padaryti pakeitimus projekto vykdymo metu, o Agile metodologijoje greitas reagavimas į pakeitimus yra vienas iš pagrindinių principų.

Dar vienas Agile bei klasikinio Krioklio principu sukurtų metodikų skirtumas yra projekto gyvavimo ciklo struktūra. Klasikinis projekto gyvavimo ciklas susideda iš inicijavimo, planavimo, vykdymo, kontrolės ir užbaigimo etapų, kurie yra vykdomi nuoseklia seka. Kalbant apie Agile metodikas, galima pastebėti, kad kai kurie projekto gyvavimo etapai sutampa ir su klasikiniu metodikos ciklu, tačiau jų seka nebėra nuosekli, kai kurie etapai gali kartotis, kiti gali būti praleisti arba įterpiami nauji projekto gyvavimo ciklo etapai arba iš viso yra atsisakoma projekto gyvavimo ciklo sampratos. [BK09]

5. PROCESO SCHEMA



Darbo užduočių sąrašas – reikalavimų, išdėstytų prioritetų tvarka, sąrašas. Po darbo užduočių sąrašo sudarymo pradedama planuoti sprintą.

Iš darbo užduočių sąrašo sprinto planavimo metu išrenkamos užduotys, kurios bus atliekamos pirmo sprinto metu. Pirmajam sprintui buvo atrinktos tokios užduotys: darbo organizavimo procesai, jų reikšmė; Agile – kas tai; Agile metodai.

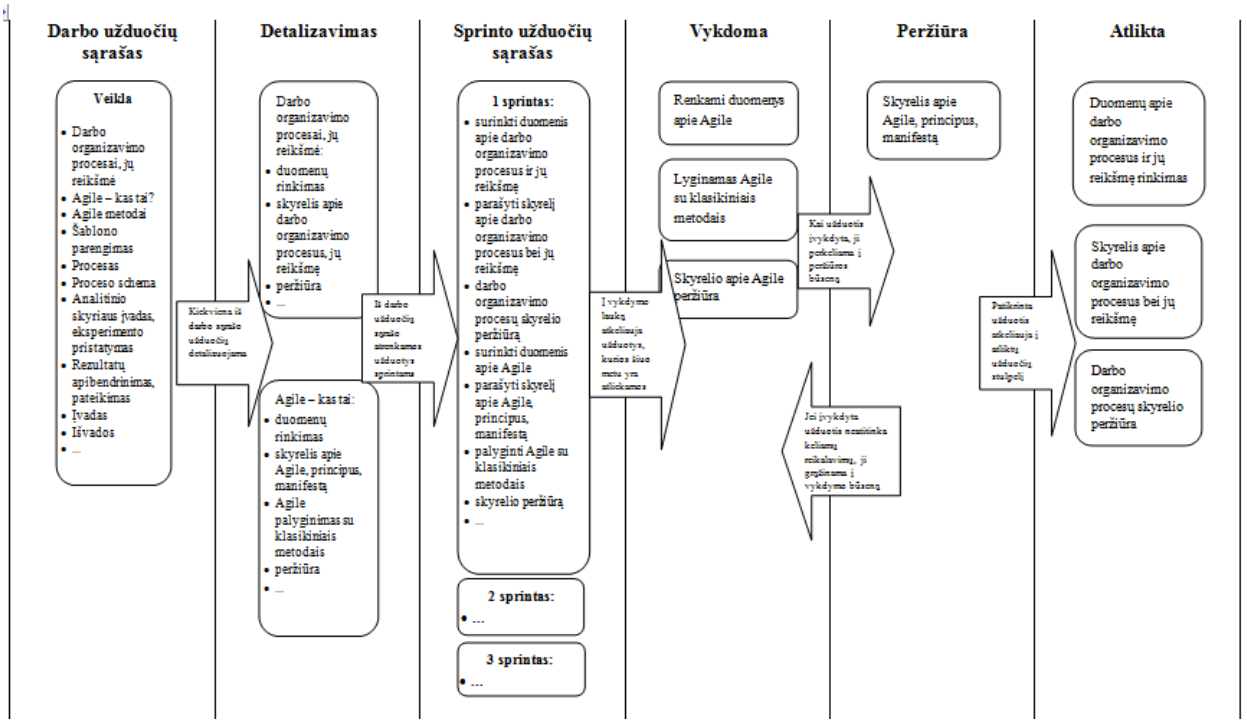
Nustatyta darbo sprinto trukmė - 3 savaitės. Taigi visas darbo laikotarpis buvo suskaidytas tokio ilgio iteracijomis.

Sprinto metu kiekvieną rytą vyko kasdieniai aptarimai. Susiskambinę 9 valandą ryto, aptardavome nuveiktus bei planuojamus darbus, problemas, su kuriomis buvo susidurta atliekant užduotis.

Po 3 savaičių vyko numatyta sprinto peržiūra. Susitikimo metu buvo aptarta, kas mūsų manymu šiame sprinte buvo teigiama, kas pavyko ne taip, kaip tikėjomės ir kaip tuos trūkumus galima būtų ištaisyti. Po įvykusios retrospektyvos jau buvo parengtas pirmas darbo skyrius.

Tada visas ciklas prasideda iš naujo – iš darbo užduočių sąrašo sprinto planavimo metu išrenkamos užduotys antrajam sprintui ir t.t.

6. KANBAN SCHEMA



Šio baigiamojo darbo Kanban lenta sudaryta iš 6 stulpelių: darbo užduočių sąrašo, detalizavimo, sprinto užduočių sąrašo, vykdymo, peržiūros bei atliktų užduočių stulpelių.

Pirmame – darbo užduočių stulpelyje išdėstytos veiklos ir užduotys, kurias reikia atlikti. Visos šios užduotys išdėstytos eilės tvarka, t.y. jos yra atliekamos pradedant aukščiausiai stulpelyje esančia užduotimi.

Į antrą stulpelį perkelta užduotis yra detalizuojama, t.y. patikslinama, ką būtent reikia padaryti arba kokio rezultato yra tikimasi atlikus šią užduotį. Pavyzdžiui, detalizuojant pirmą užduotį – darbo organizavimo procesai, jų reikšmė – buvo išskirtos svarbiausios šios užduoties dalys: surinkti duomenis apie darbo organizavimą, prašyti skyrelį apie darbo organizavimo procesus bei jų reikšmę, atlikti skyrelio peržiūrą, t.y. patikrinti ar visi užduoties punktai yra įvykdyti. Tokiu būdu yra detalizuojamos visos užduotys, esančios darbo užduočių sąraše.

Trečiame stulpelyje – sprinto užduočių sąraše – nurodomos visos detalizuotos kiekvieno sprinto užduotys bei veiklos. Jos taip pat surašytos eilės tvarka ir pradedamos vykdyti pradedant pirmąja užduotimi.

Į vykdymo stulpelį yra perkeliama ta užduotis, kuri yra šiuo metu vykdoma. Šiame stulpelyje vienu metu gali būti ir kelios užduotys. Pavyzdžiui, tuo pačiu metu galima rinkti duomenis apie Agile metodus bei lyginti juos su klasikiniais metodais.

Peržiūros stulpelyje yra užduotys, kurios yra įvykdytos, tačiau dar nepatikrintos, neperžiūrėtos. Šioje būsenoje esanti veikla patikrinama pagal visus iškeltus reikalavimus. Ir jeigu visi reikalavimai yra patenkinti, užduotis perkeliama į paskutinį - atliktų užduočių stulpelį.

Tačiau peržiūros metu galima aptikti netikslumų, nuspręsti ką nors papildyti arba, pavyzdžiui, visą skyrelį perrašyti iš naujo. Tokiu atveju užduotis vėl grąžinama į vykdymo stulpelį, yra papildoma arba ištaisomi trūkumai ir vėl perkeliama į peržiūros stulpelį. Jame užduotis patikrinama ir perkeliama į atliktų užduočių stulpelį arba vėl grąžinama atgal.

Į atliktų užduočių stulpelį atkeliauja tik tos užduotys, kurios yra patikrintos ir atitinka visus šiai užduočiai keliamus reikalavimus.

Kuo greičiau užduotis „perbėgs“ per visus lentoje esančius stulpelius, tuo greičiau bus įvykdytas projektas.

7. PROCESAS

7.1. Tvarkaraščio sudarymas

Priimant sprendimus visada iškyla jų įgyvendinimo problema, t.y. sprendimo realizavimo veiksmų planavimo arba tvarkaraščio sudarymo problema. Sprendimų priėmimo ir tvarkaraščio sudarymo procesai tarpusavyje susiję: įvairiuose planavimo etapuose tenka priimti sprendimus ir tuo pačiu metu įvairios sprendimų priėmimo procedūros reikalauja tam tikro planavimo arba detalaus tvarkaraščio sudarymo. [Pa12]

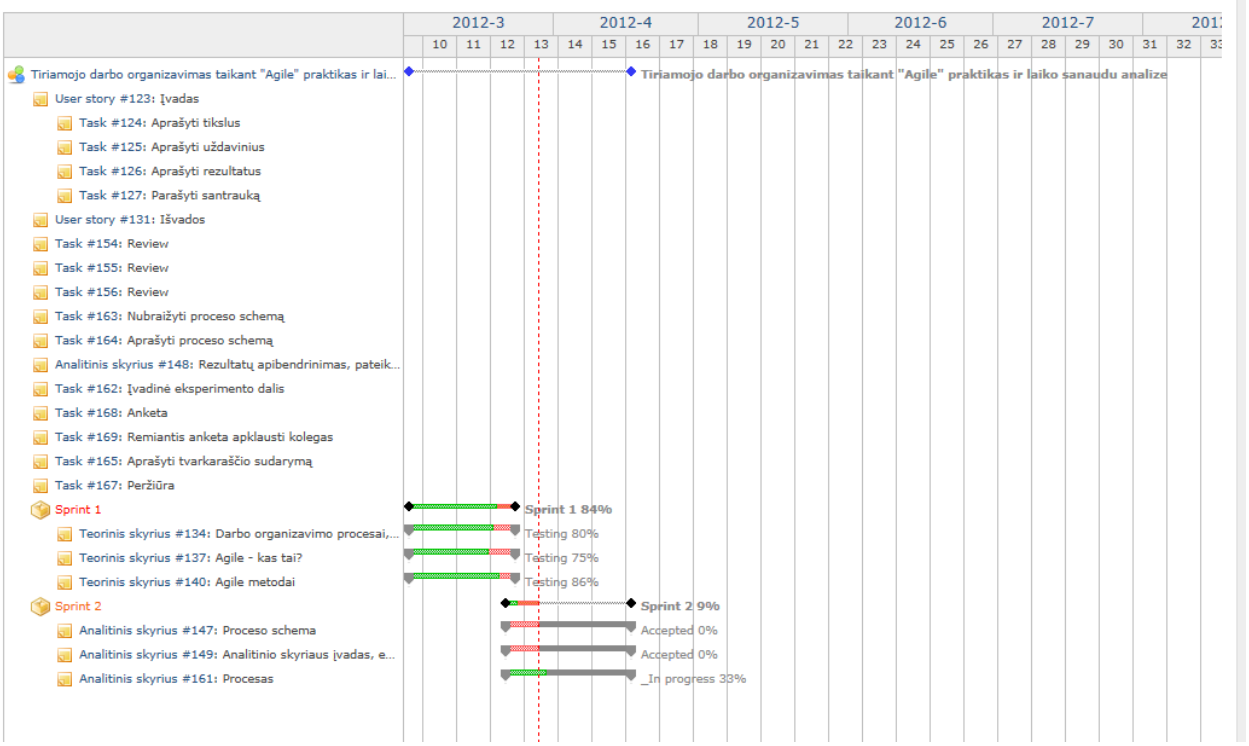
Tvarkaraščio sudarymo eigoje reikėjo numatyti, kiek laiko truks projekto sprintai ir kiekvienos sprinto užduoties atlikimas, be to reikėjo nepamiršti apie galimas rizikas, kurios galėtų sutrikdyti darbo procesą, t.y. numatyti aplinkybes, kurios galėjo sukliudyti mūsų planų įgyvendinimui. Šie ir kiti klausimai kyla kaskart realaus tvarkaraščio sudarymo procese.

Tvarkaraščio sudarymo būdų pasirinkimas gana platus. Viskas priklauso nuo projekto sudėtingumo. Juostinė planavimo technika – Gantt diagrama – koordinačių sistema, kurios kairėje pusėje išvardytos užduotys, o viršutinėje dalyje yra laiko skalė. Koordinačių tinklas yra užpildomas juostelėmis, žyminčiomis kiekvienos užduoties atlikimo trukmę. Toks būdas yra santykinai paprastas, todėl dažniausias jis yra naudojamas rengiant nedidelius, paprastesnius projektus, kai kiti planavimo būdai reikalauja per didelių laiko sąnaudų. Labai svarbu, kad turima informacija leistų įvertinti, kiek laiko reikia kiekvienai užduočiai atlikti. Gantt diagramai sudaryti galima pasinaudoti kompiuterinių programų teikiamomis galimybėmis arba specialiomis planavimo lentomis.

Tiriamojo darbo tvarkaraščio sudarymas yra vienas iš pagrindinių ir svarbiausių darbų, nuo to priklauso ar projektas bus įgyvendintas laiku, ar pavyks spėti atlikti visus užsibrėžtus tikslus ir užduotis. Tvarkaraščio sudarymas pareikalavo daug laiko, reikėjo nuosekliai apžvelgti visą darbo planą, aptarti siekiamus tikslus, realias galimybes bei numatyti rizikas. Tai gana kruopštus bei atsakingas darbas. Visus numatytus darbus išdėsčius logine seka, galima buvo imtis tvarkaraščio sudarymo – reikėjo dar kartą peržvelgti visus numatytus darbus, patikslinti ir detalizuoti kiekvieną iš jų bei nustatyti šių darbų atlikimo terminus.

Visos šio tiriamojo darbo užduotys suskaidytos sprintais, kitaip tariant, visas projektas padalintas į dalis. Vienas sprintas buvo skirtas teoriniam skyriui, kitas – analitiniam ir t.t. Po kiekvieno sprinto visada yra peržiūrimas rezultatas, nustatomi proceso trūkumai ir privalumai. Todėl visas tvarkaraštis taip pat tarsi suskaidytas dalimis, taip daug patogiau planuoti ir kontroliuoti darbo proceso eigą.

Sudarius tvarkarašti, įvertinus kiekvienos sprinto užduoties atlikimo laiką bei įvedus duomenis, buvo gauta tokią Gantt diagramą:



Visas tvarkaraštis suskaidytas sprintais, o kiekviena sprinto užduotis žymima juostele, be to pateikiama informacija apie tai, kokioje būsenoje yra kiekviena iš sprinto užduočių. Tvarkaraščio sudarymas Gannt diagramos pagalba yra patogus dar ir tuo, kad diagrama parodo kiekvienos užduoties trukmę ir atskirų užduočių santykį laike. O tai leidžia kontroliuoti darbo proceso eigą ir pamatyti grafinį projekto vaizdą.

7.2. Susitikimai

Vienas iš pagrindinių Agile metodų principų yra komunikacija ir bendravimas. Todėl kasdieniai aptarimai bei retrospektyva yra labai svarbi proceso dalis.

Kasdienių aptarimų metu yra dalijimąsi informacija bei atsakoma į tris pagrindinius klausimus:

- Kas buvo padaryta vakar?
- Kas bus padaryta šiandien?
- Su kokiomis problemomis buvo susidurta?

Šio baigiamojo darbo proceso kasdieniai aptarimai vyko telefonu kiekvieną dieną 9 valandą ryto. Pokalbio metu buvo aptariama:

- kas buvo nuveikta vakar, pavyzdžiui, ar parašytame skyrelyje aprašyti visi reikiami punktai, ar nieko netrūksta ir pan.

- kas bus daroma šiandien, t.y. sekančios sprinto užduoties aptarimas, patikslinimas, detalizavimas, ko būtent reikia ir į ką reikėtų atkreipti dėmesį.
- kas buvo neaišku, kokios informacijos pritrūko, kokios pagalbos reikėtų ir pan.

Kasdienių aptarimų nauda yra labai didelė, kadangi nuolatinis bendravimas ir problemų aptarimas neleidžia atsirasti rizikai, kad vėliau bus aptiktos rimtos klaidos, bei nenutrūksta darbo procesas – su iškylančiomis problemomis greitai susitvarkoma bendrų pastangų pagalba.

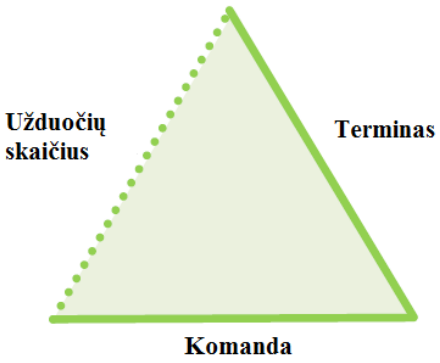
Kiekvieno sprinto pradžioje yra rengiamas sprinto planavimo susitikimas. Sprinto planavimui būtina turėti kokybišką užduočių sąrašą, t.y.:

- visi sprinto užduočių sąrašo elementai turi turėti skaitinę svarbą;
- patys svarbiausi užduočių sąrašo elementai turi būti patikslinti ir suprantami visiems komandos nariams;

Sprinto planavimas susideda iš dviejų etapų:

- pirmas etapas: nustatyti sprinto tikslą bei sprinto užduočių sąrašo funkcionalumą, kuris bus atliktas sekančio sprinto metu;
- antras etapas: nustatyti, kaip būtent turi būti atliekamos pasirinktos užduotys sprinto tikslui pasiekti. Kiekvienam sprinto užduočių sąrašo elementui pateikiamas užduočių sąrašas ir įvertinama užduočių trukmė;

Pagrindinis sprinto planavimo rezultatas – sprinto užduočių, kurias komanda planuoja realizuoti sprinto metu, sąrašas. Kadangi sprinto trukmė griežtai apribota, komanda turi nustatyti užduočių sąrašo elementų, kuriuos ji gali realizuoti, skaičių. Tokia situacija gali būti pavaizduota trikampio pavidalu:



Retrospektyva yra labai svarbi proceso dalis, ji gali trukti nuo 30 minučių iki 4 valandų, tai priklauso nuo kelių faktorių:

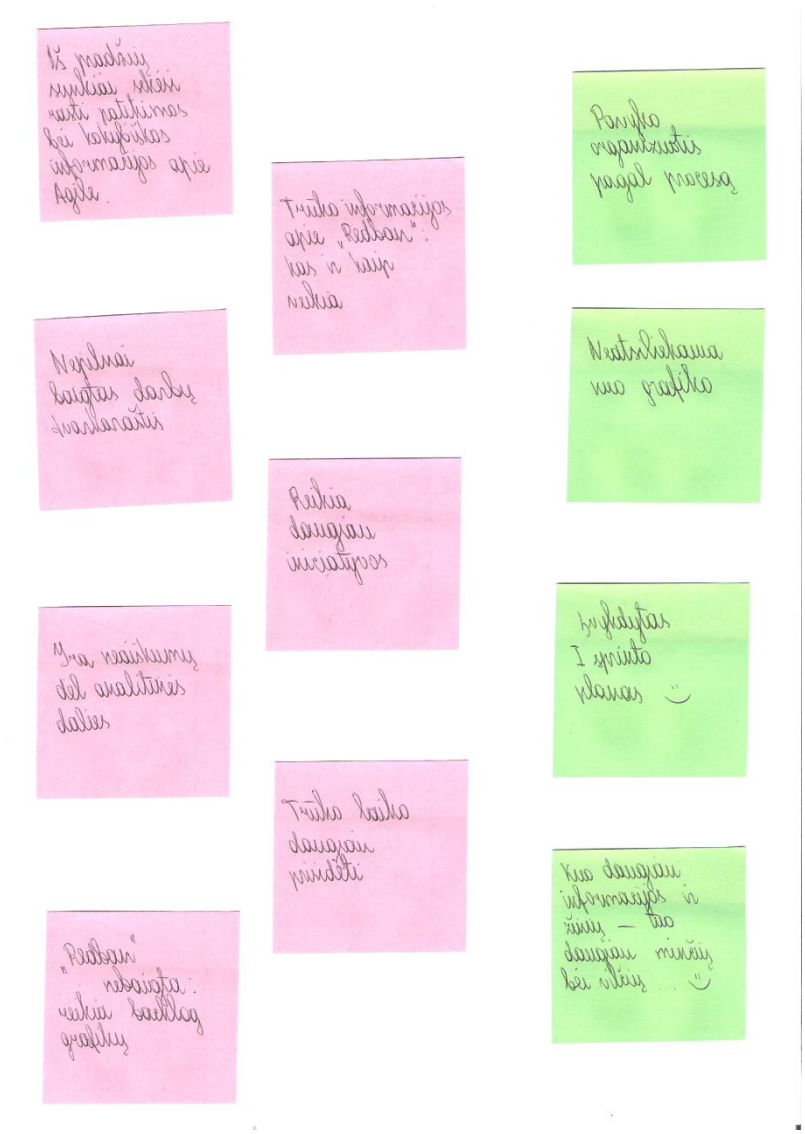
- sprinto trukmė: kuo didesnė sprinto trukmė, tuo daugiau komanda spėja padaryti ir tuo daugiau reikia aptarti;

- komandos dydis: kuo didesnė komanda, tuo daugiau reikia laiko, kad kiekvienas jos narys galėtų pasisakyti;
- problemos: laikui bėgant komanda sprendžia problemas ir retrospektyvos trukmė gali sumažėti;

Retrospektyvos metu užduodami tris pagrindiniai klausimai:

- Kas buvo padaryta gerai?
- Ką galima pagerinti?
- Kokiu būdu tai galima padaryti?

Šio baigiamojo darbo proceso retrospektyvos metu šių klausimų apgalvojimui ir atsakymams buvo skirta apie 10 minučių. Savo mintis mes reikėjo parašyti ant lipnių popieriukų ir vėliau parašytos mintys buvo aptartos. Lipnūs lapukai priklijuojami ant lapo – tai tam tikra dokumentacija. Tai, kas buvo teigiama, parašyta ant rožinių lapelių, tai, kad buvo neigiama – ant žalių:



Antroji retrospektyvos dalis – rizikų nustatymas. Realiai įvertinti galimas rizikas bei numatyti jų valdymo veiksmus yra labai svarbu, kadangi nuo to priklauso darbo proceso eiga. Retrospektyvos metu įvardytos rizikos buvo dvi, tačiau gana rimtos: nespėti iki nustatytų terminų parašyti darbo bei neatitikti baigiamajam darbui keliamus reikalavimus. Nors rizikos ir realios, tačiau buvo nuspręsta kol kas nedaryti jokių kardinalių pokyčių, juk sprinto planas įvykdytas laiku, o darbo tikslai ir užduotys (jei viskas bus įvykdyta) atitinka reikalavimus.

Trečioji retrospektyvos dalis – veiksmų planas, t.y. kas bus daroma sekančio sprinto metu. Buvo aptarta analitinė darbo dalis, kurią sudarys proceso schemos bei jų aprašymas, tvarkaraščio sudarymas, susitikimų aprašymas bei laiko įverčių analizė ir vizualizavimas.

Retrospektyva truko kiek daugiau nei valandą, buvo spėta apžvelgti visus tris minėtus retrospektyvos etapus bei nusiteikti tolimesniems darbams.

Retrospektyva yra labai naudinga darbo proceso dalis, kadangi galima pamatyti realią situaciją, peržiūrėti bei aptarti tai, kas jau yra padaryta, įvardyti visas realiai gresiančias rizikas, galimus jų valdymo metodus bei apžvelgti sekančio sprinto veiksmų planą.

7 3. Greitis

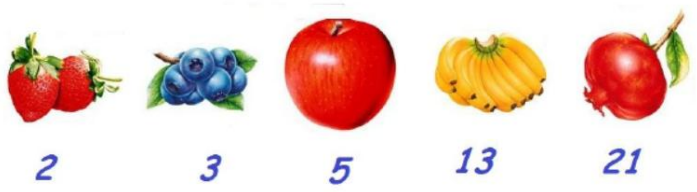
Greitis[18] - tai paprastas būdas tiksliai išmatuoti greitį, per kurį komanda spėja padaryti vienos iteracijos užsibrėžtus darbus. Turint omenyje, kad visos iteracijos vienodo ilgio (vienas iš Scrum principų), gaunama, kiek komanda realiai spėja padaryti per fiksuotą laiko intervalą.

Gana paprastai greičio sąvoką galima paaiškinti remiantis pavyzdžiu, iliustruojančiu kelis žingsnius:

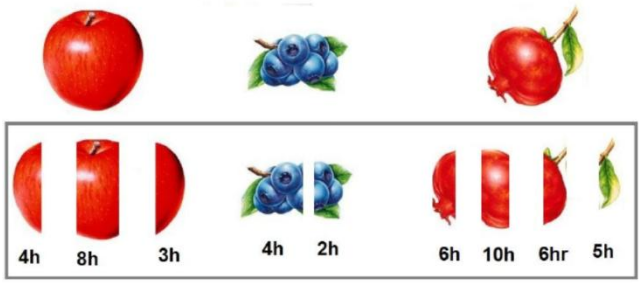
- 1 žingsnis – išskiriamos vartotojo istorijos:

- 2 žingsnis – vartotojo istorijos sudedamos eilės tvarka bei kiekvienai vartotojo istorijai suteikiamos reikšmės:

[18] Angl. *Velocity*



- 3 žingsnis – kiekviena vartotojo istorija skaidoma į smulkesnes užduotis:



- 4 žingsnis – apskaičiuojamas greitis:
    - prognozuojamas komandos greitis:
        - obuolys – 5 taškai = 15 valandų
        - mėlynės – 3 taškai = 6 valandos
        - granatas – 21 taškas = 27 valandos
    - Suma: 29 taškai = 48 valandos
    - Greitis: 1 taškas – 1,65 valandos

Komandos greitis priklauso nuo kiekvienos užduoties vertinimo proceso, kiekvieno komandos nario greičio. Būtina turėti omenyje ir tai, kad komandos greitis visą laiką keičiasi. Tai priklauso nuo komandos sudėties, užduočių tipo, komandos motyvacijos bei komandos patirties.

- 5 žingsnis – prognozuojama iteracijos trukmė:
    - apskaičiuojama visų vartotojų istorijų taškų suma: $3 + 5 + 21 + 2 + 13 = 44$ taškai;
    - apskaičiuojama visos iteracijos trukmė, atsižvelgiant į komandos greitį: $44 \cdot 1{,}65 = 72$ valados;
    - prognozuojama likusių vartotojų istorijų trukmė valandomis:
        - braškės: $2 \cdot 1{,}65 = 3$ valandos
        - bananai: $13 \cdot 1{,}65 = 22$ valandos

Pateiktas pavyzdys yra vienas iš paprasčiausių būdų, kaip galima greitai ir be sudėtingesnių skaičiavimų sužinoti tikslų komandos greitį bei panaudoti jį „burndown“ diagramos kūrimui ir darbo eigos prognozei. [IIIe11]

8. LAIKO ĮVERČIŲ ANALIZĖ IR VIZUALIZAVIMAS

8.1. Laiko įvertinimas ir vizualizavimas

Vartotojų istorijų taškai[19] gali būti apibūdinami kaip sąlyginiai vienetai, kurių pagalba įvertinama kiekviena iš iteracijos užduočių. Užduočių įvertinimas – tai komandinis darbas ir visi komandos nariai dalyvauja kiekvienos užduoties įvertinime. Kodėl?

- planavimo metu dažniausiai nėra žinoma, kas atliks vieną ar kitą užduoties dalį;
- užduočių realizavimas reikalauja įvairių specialistų dalyvavimo;
- tam, kad kiekvienas komandos narys galėtų duoti kažkokį įvertinimą, jis turi daugiau ar mažiau suprasti, kokia užduoties prasmė. Gaunant užduoties įvertinimą iš kiekvieno komandos nario, įsitikinama, kad visi supranta, apie ką eina kalba. Tai reiškia, kad užduoties atlikimo metu komandos nariai gali tikėtis vienas kito pagalbos;

Paprašius visų komandos narių įvertinti vartotojo istoriją, dažniausiai žmogus, kuris supranta užduotį geriau už kitus, paskelbs įvertinimą pirmas. Deja, tai stipriai veikia kitų komandos narių įvertinimus.

Tačiau egzistuoja praktika, kurį leidžia to išvengti. Ši praktika vadinama Pokerio planavimu[20]:

Pokerio planavimas vyksta tokiu būdu:

1. Kiekvienam komandos nariui vartotojo istorijos įvertinimui išduodama kortų kaladė su skaitmeninis ir simboliais.

[19] Angl. *Story Points*

[20] Angl. *Planning poker*

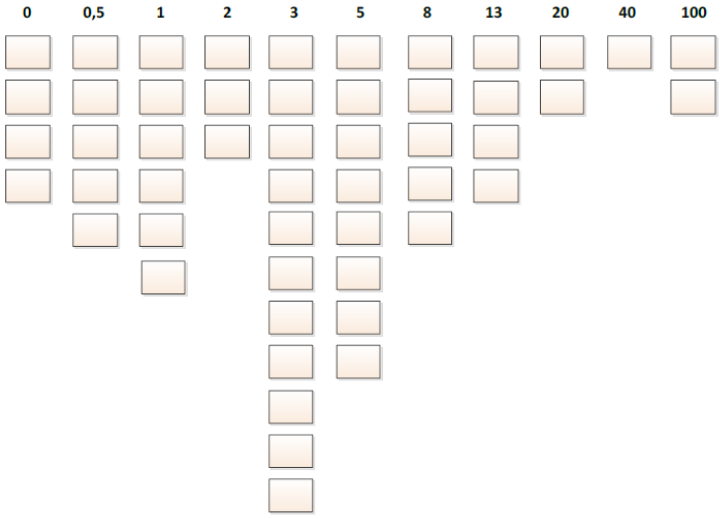
2. Pradedamas vartotojo istorijos aptarimas ir vertinimas: perskaitoma istorija, komanda užduoda klausimus, išsiaiškinamos visos detalės, jeigu tai yra būtina.
3. Kiekvienas komandos narys duoda savo įvertimą, padėdamas kortą įvertinimu žemyn.
4. Po to, kai visi komandos nariai padarė vertinimą – visos kortos apverčiamos ir vertinimai lyginami tarpusavyje.
5. Jeigu visi komandos narių vertinimai vienodi – įvertinimas priimamas, priešingu atveju pradedamas pakartotinas aptarimas ir antras raundas.

Labai svarbu suprasti, kad vertinimas turi būti vienbalsis, o ne daugumos komandos narių nuomonė.

Vartotojo istorijos vertinimui naudojama diskreti logaritminė skalė:

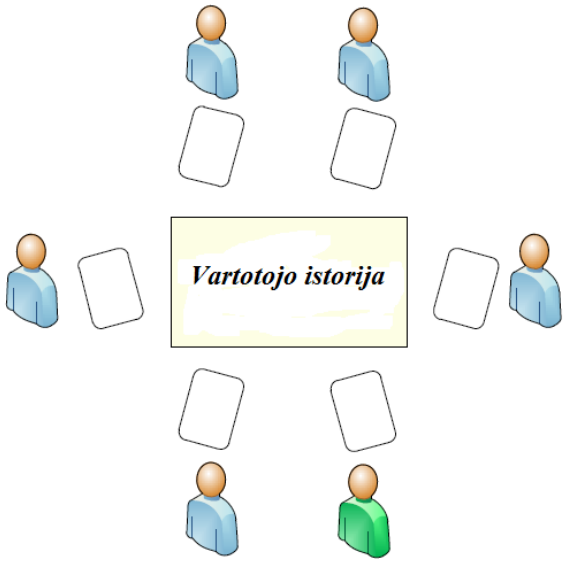
0 0,5 1 2 3 5 8 13 20 40 100

Tokiu būdu visos vartotojų istorijos darbo užduočių sąraše bus išdėstytos pagal tokias kategorijas:



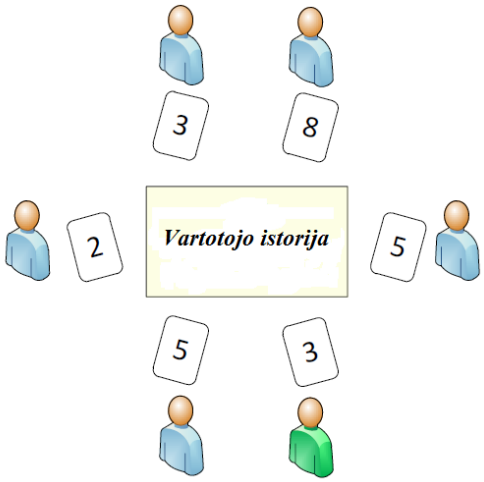
Pokerio planavimo eiga:

- Po vartotojo istorijų aptarimo pradedamas pirmas kortų dalijimas, kurio metu visi komandos nariai deda kortas vertinimu žemyn (duotoje schemoje Scrum meistras pažymėtas žaliai):

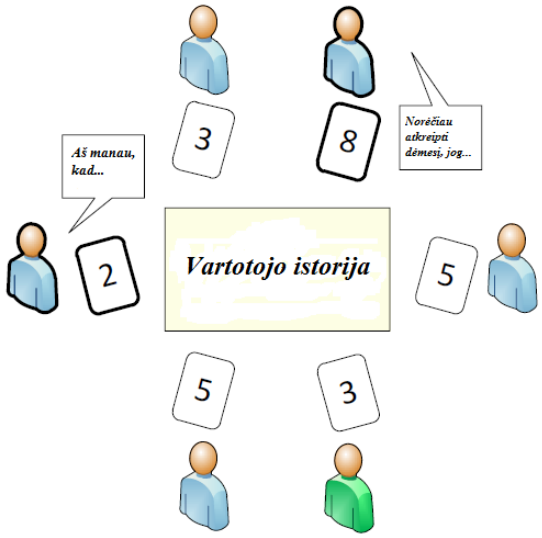


Labai svarbu, kad vartotojo istorijų aptarimų metu nebūtų spaudimo, kuriam visi nevalingai pradeda paklusti ir vertinti jau nebegalvodami apie esmę.

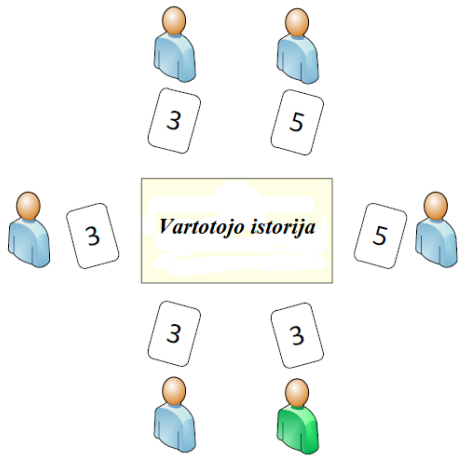
- Po to visos kortos vienu metu yra apverčiamos:



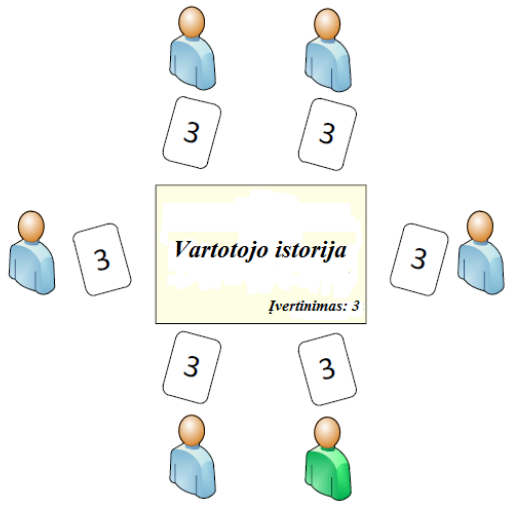
- Paskui Scrum meistras organizuoja aptarimą: galima pradėti nuo komandos narių, kurie pasirinko maksimalų arba minimalų įvertinimą:



- Scrum meistras turi sekti laiką, kuris skiriamas vartotojų istorijų aptarimui. Laikui pasibaigus pradedamas antras raundas ir sekantis aptarimas:

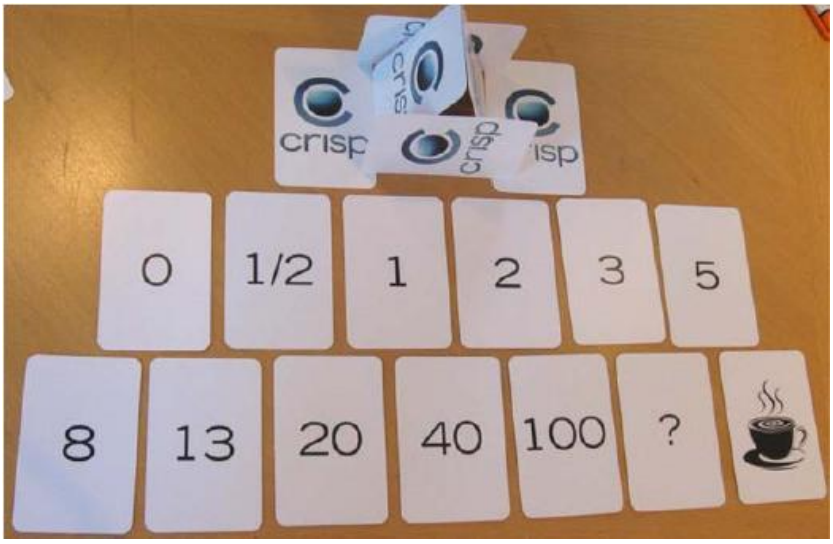


- Dažniausiai vertinimų išsibarstymas sumažėja sulig kiekvienu raundu ir pasiekiamas vienbalsis vertinimas:



Aptarimų metu tarp komandos narių išsiaiškinami visi nesutarimai dėl vartotojo istorijos vertinimo, pradedama suprasti, koks yra darbo procesas, o komanda tiksliau ir detaliau supranta užduoties esmę.

Pokerio planavimo kortų kaladėje yra kelios specialios kortos:



- 0 = „vartotojo istorija jau atlikta“ arba „kelios minutės darbo“;
- ? = „Neturiu supratimo...“;
- Kavos puodelis = „Pernelyg pavargau, kad galėčiau galvoti. Padarykime pertrauką“.

Galimas dar vienas „vertinimo žaidimas“ su paprastesnėmis taisyklėmis.

Pasiruošimas: vartotojo istorijos, kurias ruošiamasi vertinti, atspausdinamos arba tiesiog rašomos ranka ant kortelių.

Žaidimo metu kiekvienas žaidėjas gali padaryti tokius veiksmus:

- paimti vartotojo istorijos kortelę iš krūvelės viršaus ir padėti ją kur nors ant stalo atsižvelgiant į kitų kortelių padėtį. Kairėje – mažesnę vertę turinčios vartotojo istorijos, dešinėje – didesnę, ant stalo jau esamų kortelių apačioje – tokios pat vertės vartotojo istorijos.
- pastumti vieną iš ant stalo esamų kortelių, paaiškinant kodėl žaidėjas nesutinka su jos esama padėtimi.
- praleisti ėjimą.

Žaidimas baigiamas, kai kortelių krūvelėje nebelieka kortelių ir nei vienas iš žaidėjų nenori perstumti ant stalo padėtų kortelių.

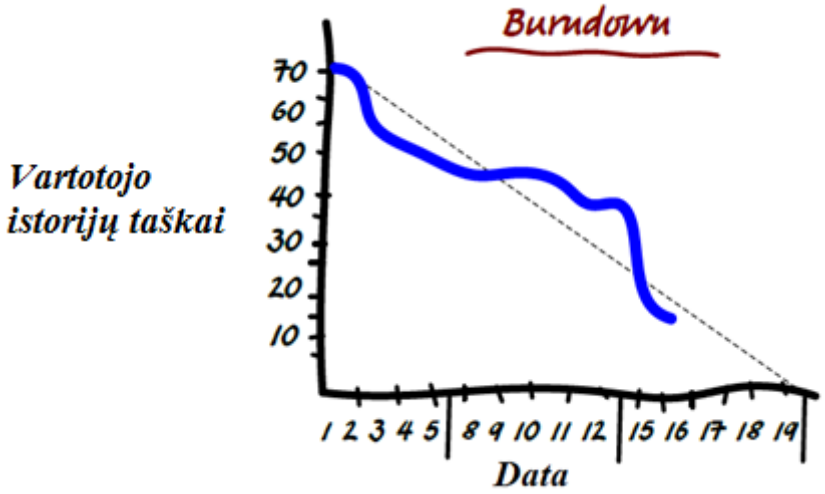
Labai svarbu bendrauti žaidimo metu ir aiškinti, kodėl žaidėjas priėmė vieną arba kitą sprendimą. Žaidimo tikslas – duoti vienbalsius komandinius vertinimus kiekvienai vartotojo istorijai.

Po to, kai ant stalo atsiras kortelių, turinčių nuo mažiausios iki didžiausios vertės, linija, komandai belieka pasirinkti vertinimo skalę. Galima naudoti skaičius nuo 1 iki 10, logaritminę skalę (0, 0,5, 1, 2, 3, 5, 8, 13, 20, 40, 100) arba Fibonačio skalę (1, 2, 3, 5, 8, 13).

8.3. „Burndown“ diagramos

Labai svarbu, kad komanda įvertintų darbų, kuriuos reikia padaryti tam, kad užduotis būtų atlikta, apimtį. Viena iš pagrindinių Agile projektų valdymo technologijų yra „burn“ diagramos. Jose pavaizduojami vartotojo istorijų įvertinimai. Yra 2 pagrindiniai „burn“ diagramų tipai: „burndown“ ir „burnup“ diagramos.

„Burndown“ diagrama – vienas iš įrankių, padedantis sekti, kiek darbų dar reikia atlikti einamoje iteracijoje. Diagramos Y – ko ašyje atidedamas vartotojų istorijų, kurias dar reikia atlikti, taškų skaičius, X – o ašyje – data arba iteracijos dienų skaičius. Papildomai konstruojama ideali kreivė (brūkšninė linija grafiko įstrižainėje), kuri rodo suplanuotą darbo eigą:



Pirmas taškas diagramoje pirmąja iteracijos dieną žymi, kiek vartotojų istorijų iš sprinto užduočių sąrašo reikia atlikti. Kiekvieną dieną grafike atsiranda naujas taškas, kuris rodo, kiek neatliktų darbų liko sprinto užduočių sąraše.

Jeigu nėra stipriai atsiliekama nuo suplanuoto grafiko arba jis nėra stipriai lenkiamas, „burndown“ diagramos kreivė turi apytiksliai sutapti su sukonstruota idealia kreive.

„Burndown“ diagramos analizė atliekama lyginant realią kreivę su idealia:

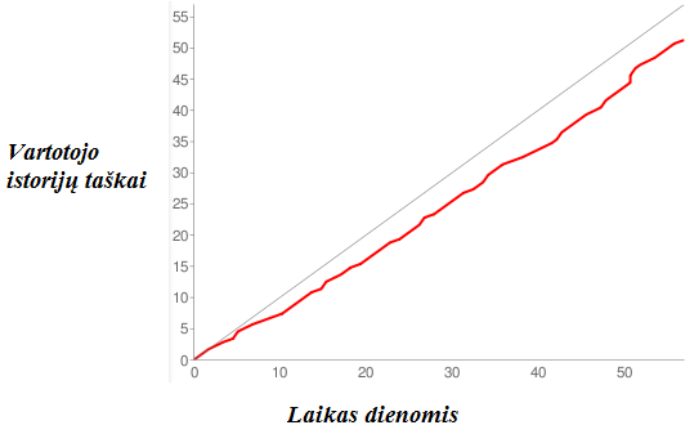
- jeigu reali kreivė yra aukščiau idealios, reiškia, komanda atsilieka nuo plano;
- jeigu reali kreivė yra žemiau idealios, reiškia, komanda lenkia planą.

Diagramos privalumai:

- yra matoma visiems,
- skirta komandos nariams, o ne vadovams,
- analizuojama kiekvieną dieną,
- komanda priima sprendimus, kai diagramos rodikliai yra maži arba dideli.

Pagrindinė „borndown“ diagramos paskirtis – kaip galima anksčiau nustatyti atsilikimą nuo grafiko ar, atvirkščiai, jo lenkimą, kad būtų galimybė tinkamai sureaguoti.

„Burnup“ diagramoje vaizduojama proceso eiga. Y – ko ašyje atidedamas atliktų vartotojų istorijų taškų skaičius, X – o ašyje – data arba iteracijos dienų skaičius. Papildomai konstruojama ideali kreivė (brūkšninė linija grafiko įstrižainėje), kuri rodo suplanuotą darbo eigą:

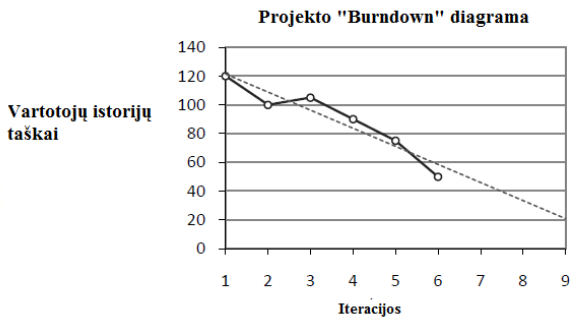


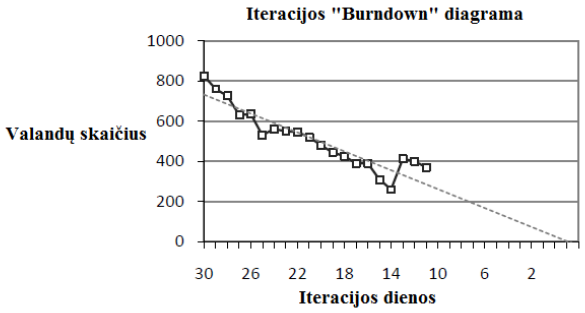
8.2. „Burndown“ diagramų tipai

Scrum projektuose proceso pažanga stebima ir fiksuojama „burndown“ diagramų pagalba. Diagramos vadinamos „burndown“, nes jos rodo, kokį darbą dar reikia padaryti, o ne koks darbas jau yra užbaigtas. „Burndown“ diagramos yra dviejų tipų:

- projekto diagrama
- iteracijos diagrama

Projekto diagrama naudojama norint stebėti bendrą projekto pažangą ir pranešti apie rezultatus tiek projekto rėmėjams, tiek komandos nariams. Projekto diagrama leidžia stebėti du pagrindinius rodiklius: bendrą proceso pažangą bei likusių darbų apimtį. Stebint bendrą proceso pažangą galima prognozuoti apytikslį projekto užbaigimo terminą. Jei į projektą įtraukiama nauja užduotis, diagramos kreivė „pakyla į viršų“, jei užduotis išimama – atitinkamai „nusileidžia žemyn“:

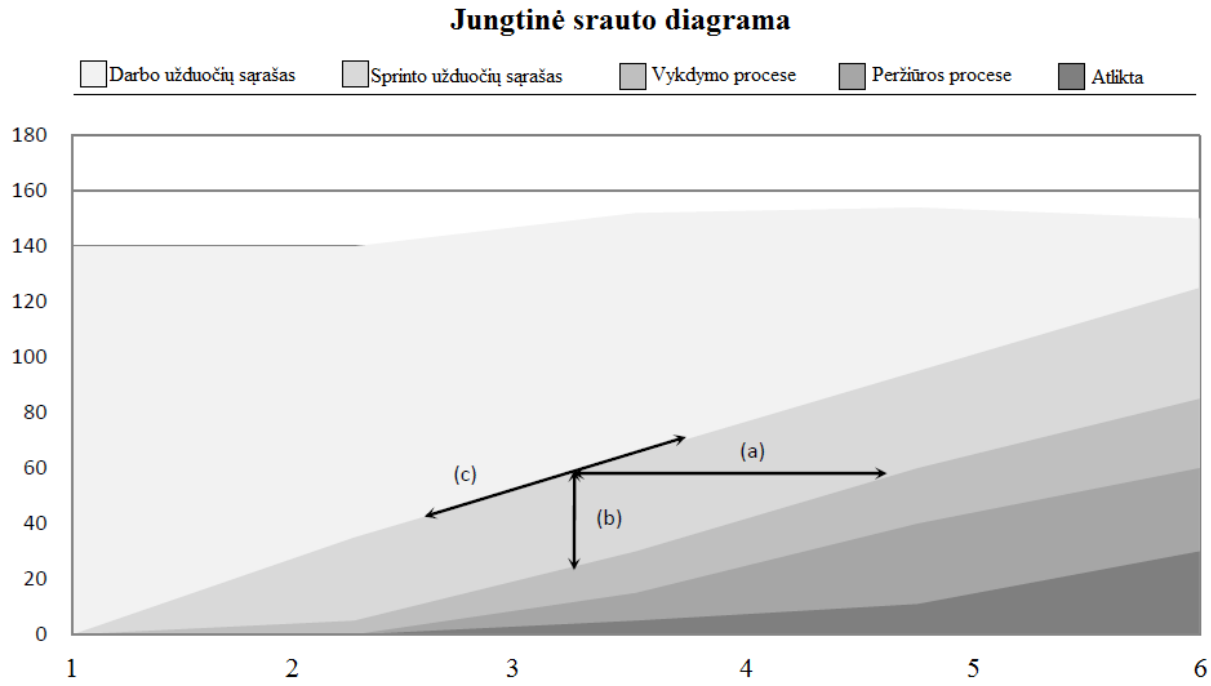




Iteracijos diagrama gaunama remiantis užduočių lentos (Kanban lentos) informacijos duomenimis ir yra skirta komandos nariams. Diagramos tikslas yra vaizdžiai pateikti informaciją apie einamosios iteracijos dienų skaičių ir užduotims skirtą valandų skaičių. Komanda iš šios diagramos gautą informaciją naudoja tam, kad galėtų stebėti, ar visos iteracijos užduotys bus baigtos numatytu laiku, ar prireiks kelių papildomų valandų arba kai kurios užduotys turės būti pakeistos arba perkeltos į kitą iteraciją. [EM09]

8.3. Jungtinės srauto diagramos

Jungtinė srauto diagrama [21] vaizduojama remiantis vartotojų istorijų, kurios jau pasiekė tam tikrą atlikimo lygį konkrečiu momentu, skaičiumi.

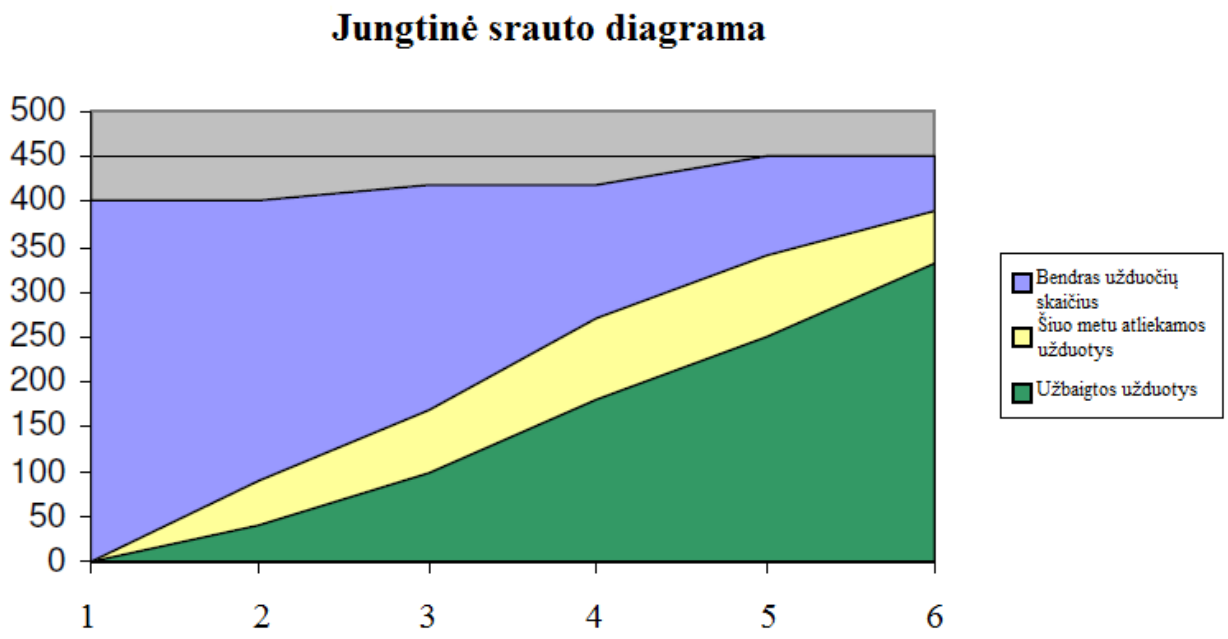


Palyginus su „Burndown“ diagramomis, jungtinė srauto diagrama pateikia daugiau informacijos: srauto rodikliai, pažangos dydis ir laikas.

Jungtinės srauto diagramos darbo užduočių sąrašo dalis rodo bendrą darbo, kurį reikia atlikti, apimtį. Šio ploto viršutinės linijos „pakilimas aukštyn“ arba „nusileidimas žemyn“ atitinkamai reiškia pridėtas arba išimtas iš projekto užduotis. Horizontali linija (a) matuoja vidutinį kiekvienos vartotojo istorijos atlikimo laiką. Vertikali linija (b) reiškia vartotojų istorijų skaičių tam tikroje tam tikru laiku ir tam tikroje būsenoje. Linija (c) nurodo rodiklį, pagal kurį vartotojo istorijos pasiekia tam tikrą būseną. [EM09]

Jungtinės srauto diagramos pavyzdys:

[21] Angl. *Cumulative Flow Chart*



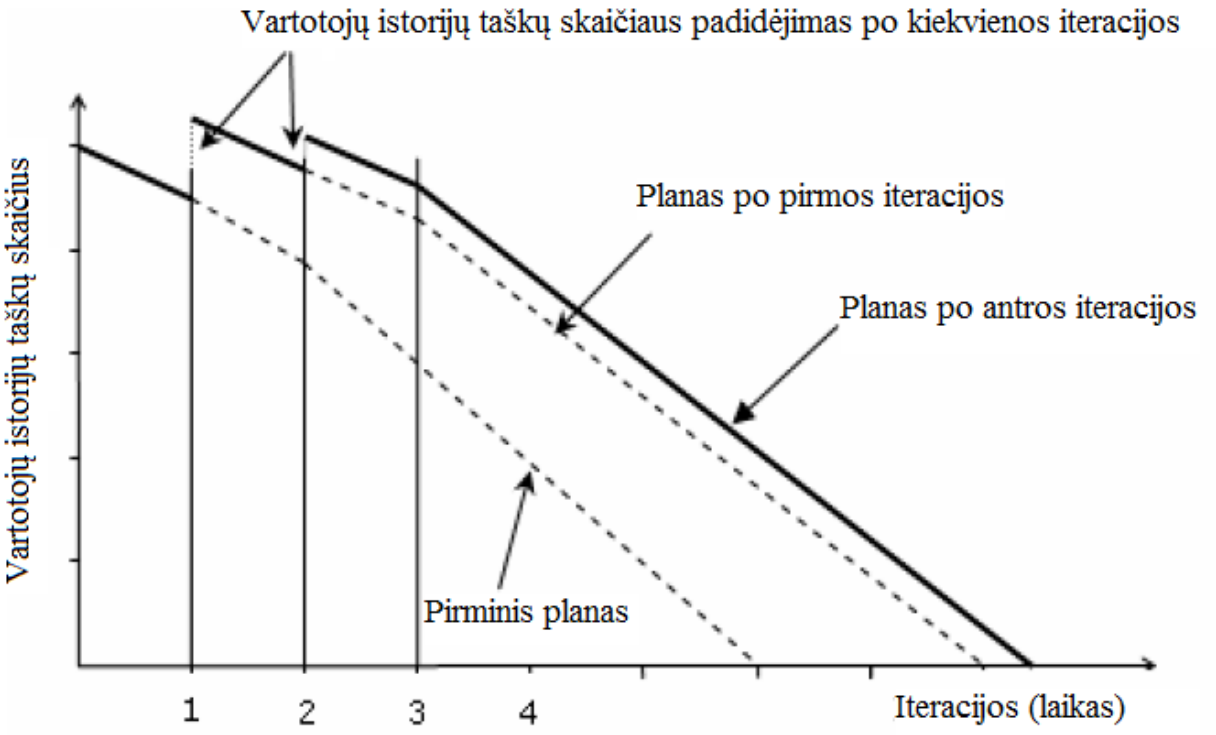
Šioje diagramoje mėlynas plotas žymi visus suplanuotus darbus, kurie turi būti atlikti. Geltonas plotas žymi visas užduotis, kurios šiuo metu yra atliekamos. Žalias plotas parodo bendrą užbaigtų užduočių skaičių.

Jungtinės srauto diagramos yra kiek sudėtingesnės nei „Burndown“ diagramos, tačiau turi pranašumą, leidžiantį vaizduoti skirtingas projekto sritis atskirai nuo bendro darbo produktyvumo. [CG12]

8.4. „Burndown“ diagramų vaizdavimo problemos

Projekto procesas ne visada vyksta tiksliai taip, kaip buvo suplanuota. Iteracijos metu ne visada pavyksta atlikti visas suplanuotas užduotis ir jas tenka įtraukti į kitos iteracijos planuojamų užduočių sąrašą arba visai atsisakyti. Galima ir tokia situacija, kai visos suplanuojamos užduotys atliekamos greičiau, nei buvo numatyta ir tenka įtraukti papildomas užduotis.

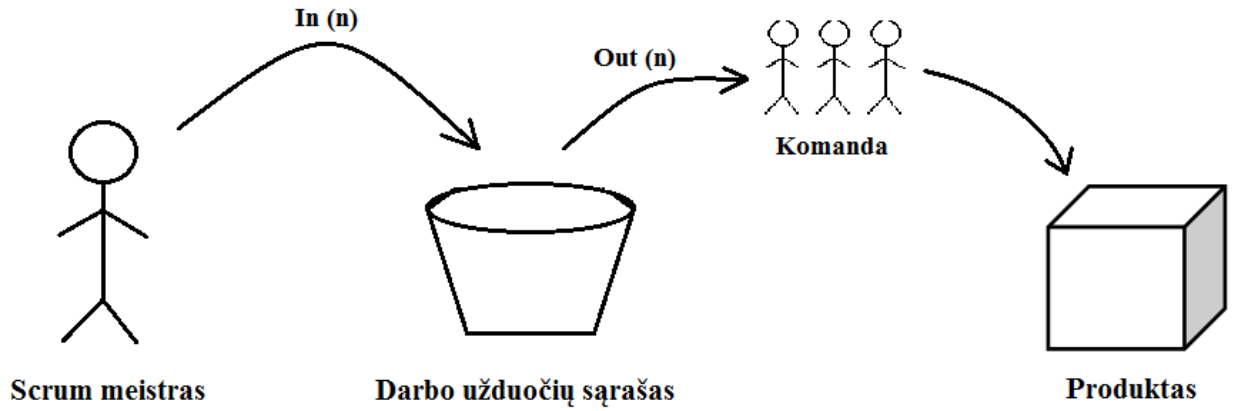
„Burndown“ diagrama fiksuoja visus užduočių kiekių pokyčius proceso metu.



Šiame paveikslėlyje pateiktas pavyzdys, kai iteracijos metu pridedamos papildomos užduotys ir tokiu būdu padidėja gautų vartotojų istorijų taškų skaičius ir, atitinkamai, pailgėja iteracijų atlikimo laikas. Tokiu būdu „burndown“ diagramos kreivė nukrypsta nuo suplanuotos kreivės, t.y. pirminio plano, todėl planuota tiesė tampa kreive ir atsiranda projekto užbaigimo laiko problema – tampa sunkau sudaryti prognozes.

8.5. Projekto užbaigimo laikotarpio prognozės sudarymas ir vizualizavimas

Projekto užbaigimo laikotarpio prognozės sudarymui bus nagrinėjamas toks modelis:



Čia:

In (n) reiškia, kiek pridėta vartotojo istorijų taškų,

Out (n) – keik vartotojų taškų yra realizuota,

n – iteracijos numeris.

In (n) ir Out (n) yra atsitiktiniai normalieji dydžiai, pasiskirstę pagal normalųjį dėsnį. Todėl šių dydžių tankio funkcijos yra išreiškiamos tokiomis formulėmis:

$$F_{in}(x) = \frac{1}{\sigma_{in}\sqrt{\pi}} e^{-\frac{(x-\overline{x_{in}})^2}{\sigma_{in}^2}}, F_{out}(x) = \frac{1}{\sigma_{out}\sqrt{\pi}} e^{-\frac{(x-\overline{x_{out}})^2}{\sigma_{out}^2}}.$$ [BK07]

Tarkime, yra įvykdyta N iteracijų ir kiekvienai iš jų mes žinome v_n ir b_n (čia v_n - n-osios iteracijos greitis, b_n – darbo užduočių sąrašo dydis n-osios iteracijos pabaigoje). Tada

$$\overline{x_{in}} = \frac{1}{N}\sum_{n=1}^{N}(v_n - b_{n-1} + b_n) \quad (1)$$

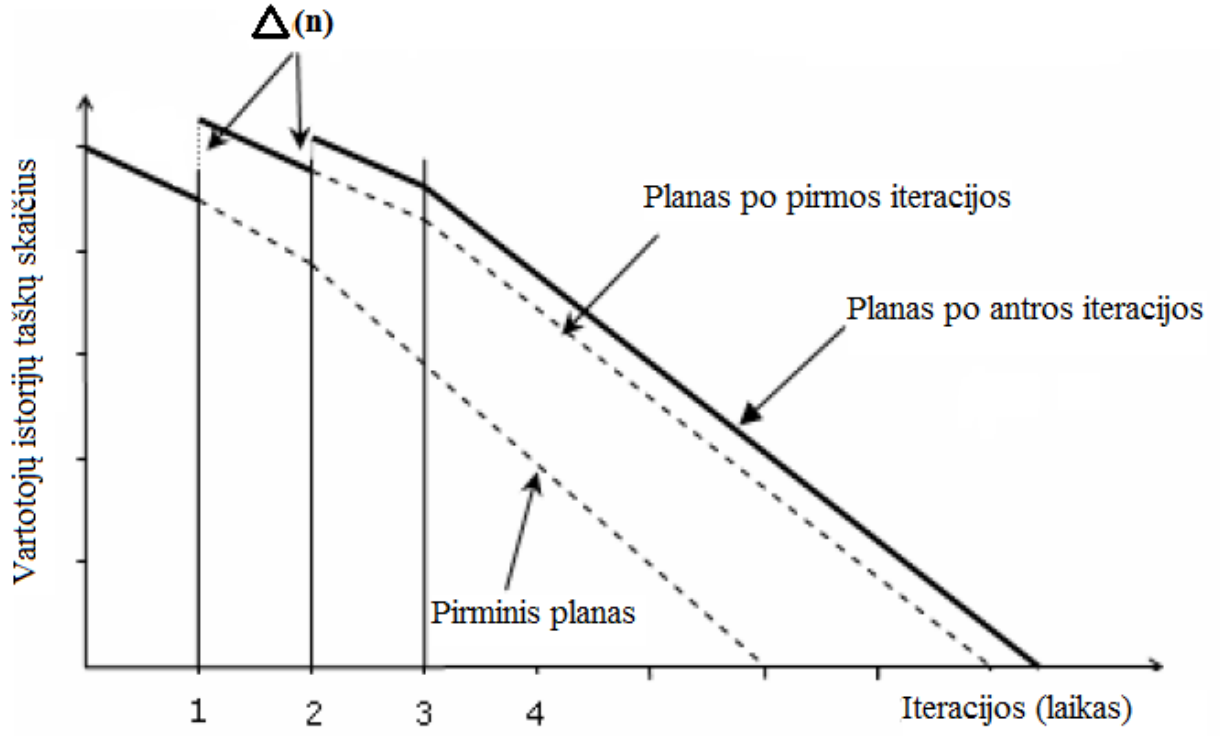
$$\sigma_{in} = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(v_n - b_{n-1} + b_n - \overline{x_{in}})^2} \quad (2)$$

$$\overline{x_{out}} = \frac{1}{N}\sum_{n=1}^{N} v_n, \quad (3)$$

$$\sigma_{out} = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(v_n - \overline{x_{out}})^2} \quad (4)$$

čia b_0 - darbo užduočių sąrašo dydis projekto pradžioje.

Darbo užduočių sąrašo pasikeitimas – taip pat atsitiktinis dydis: $\Delta(n) = In(n) - Out(n)$, $\Delta(n)$ taip pat pasiskirstęs pagal normalųjį dėsnį. [Ze12] Grafiškai vaizduojant, $\Delta(n)$ atrodytų taip:



Darbo užduočių sąrašo pasikeitimo per vieną iteraciją tankio funkciją užrašoma tokia formule:

$$F_{\Delta}^{1}(x) = \frac{1}{\sigma_{\Delta_1}\sqrt{\pi}} e^{-\frac{(x-\overline{\Delta_1})^2}{\sigma_{\Delta_1}^2}},$$

čia $\overline{\Delta_1} = \overline{x_{out}} - \overline{x_{in}}, \sigma_{\Delta_1} = \sqrt{\sigma_{in}^2 + \sigma_{out}^2}$. (5)

Darbo užduočių sąrašo pasikeitimo per M iteracijų tankio funkcija atrodo taip:

$$F_{\Delta}^{M}(x) = \frac{1}{\sigma_{\Delta_M}\sqrt{\pi}} e^{-\frac{(x-M\overline{\Delta_1})^2}{\sigma_{\Delta_M}^2}},$$

čia $\sigma_{\Delta_M} = \sqrt{M}\sigma_{\Delta_1}, \overline{\Delta_M} = M\overline{\Delta_1}$.

Arba $F_{\Delta}^{M}(x) = \frac{1}{\sigma_{\Delta_1}\sqrt{\pi M}} e^{-\frac{(x-M\overline{\Delta_1})^2}{M\sigma_{\Delta_1}^2}}$. [ČM06]

Tarkime, kad yra įvykdyta N iteracijų ir yra žinomi parametrai $\overline{\Delta_1}, \sigma_{\Delta_1}$. Kokia tikimybė, kad projektas bus sėkmingai pabaigtas, t.y. bus atliktos visos užduotys iš darbų sąrašo per M iteracijų?

Į turimą funkciją $F_{\Delta}^{M}(x)$ reikia įstatyti x – vartotojo istorijų taškų skaičių. Tokiu būdu gaunama tikimybė, kad per M iteracijų tie taškai bus „uždirbti“. T.y. reikia rasti tikimybę, kad x (su pasiskirstymo tankio funkcija $F_{\Delta}^{M}(x)$) bus didesnis nei b_n (darbo užduočių sąrašo dydis n-osios iteracijos pabaigoje). Ši tikimybė pažymima $D_N(M)$.

$D_N(M) = \int_{b_N}^{\infty} F_{\Delta}^{M}(x)dx$ arba $D_N(M) = \frac{1}{\sqrt{\pi}} \int_{\frac{b_N - M\overline{\Delta_1}}{\sigma_{\Delta_1}\sqrt{M}}}^{\infty} e^{-x^2}\, dx$. Šitą integralą galima apskaičiuoti skaičiuoklės pagalba. Skaičiuoklė turi funkciją ERFC: $ERFC(x) = \frac{2}{\sqrt{\pi}} \int_{x}^{\infty} e^{-t^2}\, dt = 1 - ERF(x)$. Išraiška $\frac{b_N - M\overline{\Delta_1}}{\sigma_{\Delta_1}\sqrt{M}}$ gali būti ir neigiama. Taigi galimi du atvejai:

a) Jeigu $(b_N - M\overline{\Delta_1}) > 0$, tai $D_N(M) = \frac{1}{2} ERFC(\frac{b_N - M\overline{\Delta_1}}{\sigma_{\Delta_1}\sqrt{M}})$;

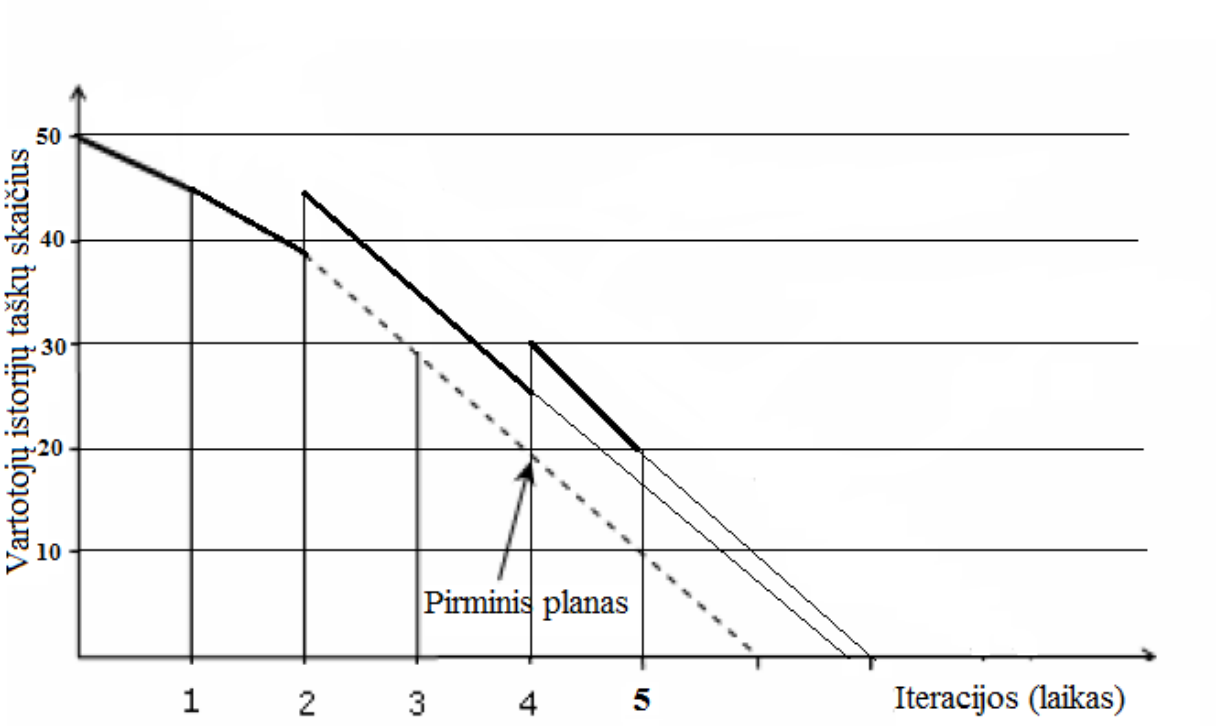
b) Jeigu $(b_N - M\overline{\Delta_1}) < 0$, tai $D_N(M) = \frac{1}{2} ERFC\left(\frac{M\overline{\Delta_1} - b_N}{\sigma_{\Delta_1}\sqrt{M}}\right) + ERF\left(\frac{M\overline{\Delta_1} - b_N}{\sigma_{\Delta_1}\sqrt{M}}\right)$.

Apibendrinimas:

1. Matuojami v_n, b_n dydžiai pirmoms N iteracijoms.
2. Skaičiuojama b_N - kiek liko „neuždirbtų“ vartotojo istorijų taškų.
3. Apskaičiuojama $\overline{\Delta_1}, \sigma_{\Delta_1}$.
4. Kiekvienam M iš intervalo $[(N+1), \dots, (kiek\ reikia)]$ skaičiuojama $D_N(M)$. [Ze12]

8.6. Atvejų analizė

Tarkime yra vykdomas projektas, kurio „burndown“ diagrama po 5 įvykdytų iteracijų atrodo taip:



Projekto metu buvo atlikti keli pakeitimai darbo užduočių sąraše, t.y. trečios ir penktos iteracijų eigoje buvo pridėtos papildomos užduotys, kurios vertinamos dešimčia vartotojo istorijų tašku. Po tokių pakeitimų atitinkamai pasikeitė ir projekto atlikimo terminai. Ankstesniame skyrelyje aprašytu būdu galima apskaičiuoti šio projekto apytikslius pabaigimo terminus, t.y. rasti tikimybę pabaigti šį projektą per tam tikrą skaičių iteracijų. Šiuo tikslu pirmiausią surašomi turimi duomenis:

$v_1 = 5, v_2 = 10, v_3 = 10, v_4 = 5, v_5 = 5$ – pirmų penkių atliktų iteracijų komandos greitis,

$b_0 = 50, b_1 = 45, b_2 = 45, b_3 = 35, b_4 = 35, b_5 = 20$ - darbo užduočių sąrašo dydis kiekvienos iš pirmų penkių įvykdytų iteracijų pabaigoje.

Remiantis (1), (2), (3) ir (4) formulėmis apskaičiuojamos $\overline{x_{in}}$, σ_{in}, $\overline{x_{out}}$ ir σ_{out} išraiškos:

$$\overline{x_{in}} = \frac{1}{5} \cdot 5 = 1, \sigma_{in} \approx 7{,}4162, \overline{x_{out}} = \frac{1}{5} \cdot 35 = 7, \sigma_{out} \approx 2.7386 \quad (6)$$

Pasinaudojus (5) formulėmis pagal turimus duomenis jau galima rasti $\overline{\Delta_1}$ bei σ_{Δ_1} išraiška:

$$\overline{\Delta_1} = 7 - 1 = 6, \sigma_{\Delta_1} \approx 7{,}9057 \quad (7)$$

Turimų duomenų pakanka kiekvienam M iš intervalo $[(N+1), \dots, (kiek\ reikia)]$ tikimybės $D_N(M)$ radimui. Jau yra užbaigtos penkios projekto iteracijos, todėl tikslinga skaičiuoti tikimybę užbaigti projektą per šešias, septynias, aštuonias, devynias bei, tarkime, dešimt iteracijų. Tačiau bus apskaičiuotos visos dešimties iteracijų tikimybės.

Iš pradžių svarbu nustatyti išraiškos $(b_N - M\overline{\Delta_1})$ ženklą. Šiuo tikslu patogu sudaryti lentelę:

Iteracijos numeris	Išraiškos$(b_N - M\overline{\Delta_1})$ rezultatas	Formulė, pagal kurią bus skaičiuojama tikimybė
1	$20 - 1 \cdot 6 = 14 > 0$	$D_5(1) = \frac{1}{2} ERFC(\frac{b_5 - \overline{\Delta_1}}{\sigma_{\Delta_1}\sqrt{1}})$;
2	$20 - 2 \cdot 6 = 8 > 0$	$D_5(2) = \frac{1}{2} ERFC(\frac{b_5 - 2\overline{\Delta_1}}{\sigma_{\Delta_1}\sqrt{2}})$;
3	$20 - 3 \cdot 6 = 2 > 0$	$D_5(3) = \frac{1}{2} ERFC(\frac{b_5 - 3\overline{\Delta_1}}{\sigma_{\Delta_1}\sqrt{3}})$;
4	$20 - 4 \cdot 6 = -4 < 0$	$D_5(4) = \frac{1}{2} ERFC\left(\frac{4\overline{\Delta_1} - b_5}{\sigma_{\Delta_1}\sqrt{4}}\right) + ERF\left(\frac{4\overline{\Delta_1} - b_5}{\sigma_{\Delta_1}\sqrt{4}}\right)$
5	$20 - 5 \cdot 6 = -10 < 0$	$D_5(5) = \frac{1}{2} ERFC\left(\frac{5\overline{\Delta_1} - b_5}{\sigma_{\Delta_1}\sqrt{5}}\right) + ERF\left(\frac{5\overline{\Delta_1} - b_5}{\sigma_{\Delta_1}\sqrt{5}}\right)$
6	$20 - 6 \cdot 6 = -16 < 0$	$D_5(6) = \frac{1}{2} ERFC\left(\frac{6\overline{\Delta_1} - b_5}{\sigma_{\Delta_1}\sqrt{6}}\right) + ERF\left(\frac{6\overline{\Delta_1} - b_5}{\sigma_{\Delta_1}\sqrt{6}}\right)$
7	$20 - 7 \cdot 6 = -22 < 0$	$D_5(7) = \frac{1}{2} ERFC\left(\frac{7\overline{\Delta_1} - b_5}{\sigma_{\Delta_1}\sqrt{7}}\right) + ERF\left(\frac{7\overline{\Delta_1} - b_5}{\sigma_{\Delta_1}\sqrt{7}}\right)$
8	$20 - 8 \cdot 6 = -28 < 0$	$D_5(8) = \frac{1}{2} ERFC\left(\frac{8\overline{\Delta_1} - b_5}{\sigma_{\Delta_1}\sqrt{8}}\right) + ERF\left(\frac{8\overline{\Delta_1} - b_5}{\sigma_{\Delta_1}\sqrt{8}}\right)$
9	$20 - 9 \cdot 6 = -34 < 0$	$D_5(9) = \frac{1}{2} ERFC\left(\frac{9\overline{\Delta_1} - b_5}{\sigma_{\Delta_1}\sqrt{9}}\right) + ERF\left(\frac{9\overline{\Delta_1} - b_5}{\sigma_{\Delta_1}\sqrt{9}}\right)$
10	$20 - 10 \cdot 6 = -40 < 0$	$D_5(10) = \frac{1}{2} ERFC\left(\frac{10\overline{\Delta_1} - b_5}{\sigma_{\Delta_1}\sqrt{10}}\right) + ERF\left(\frac{10\overline{\Delta_1} - b_5}{\sigma_{\Delta_1}\sqrt{10}}\right)$

Visų duomenų rinkimui bei rezultatų skaičiavimams buvo naudojamasi skaičiuokle, joje sudarytos lentelės su gautais rezultatais atrodo taip:

n	v_n	b_n	$v_n - b_{n-1} + b_n$	X_{in}	X_{out}	$\overline{\Delta_1}$	$(v_n - b_{n-1} + b_n - X_{in})^2$
0	0	50	-	1	7	6	-
1	5	45	0				1
2	10	45	10				81
3	10	35	0				1
4	5	35	5				16
5	5	20	-10				121

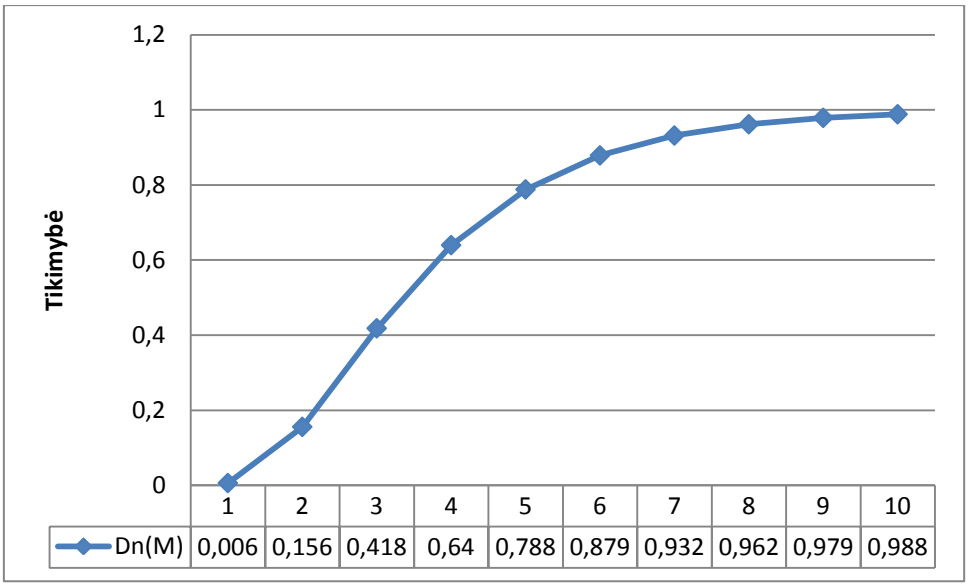
n	σ_{in}	$(v_n - X_{out})^2$	σ_{out}	σ_{Δ_1}
0	7,416198487	-	2,738612788	7,90569415
1		4		
2		9		
3		9		
4		4		
5		4		

M	$b_N - M\overline{\Delta_1}$	$M\overline{\Delta_1} - b_N$	$\sigma_{\Delta_1}\sqrt{M}$	$\frac{b_N - M\overline{\Delta_1}}{\sigma_{\Delta_1}\sqrt{M}}$	$\frac{M\overline{\Delta_1} - b_N}{\sigma_{\Delta_1}\sqrt{M}}$
1	14	-	7,90569415	1,77087549	-
2	8	-	11,18033989	0,715541753	-
3	2	-	13,69306394	0,146059349	-
4	-4	4	15,8113883	-	0,252982213
5	-10	10	17,67766953	-	0,565685425
6	-16	16	19,36491673	-	0,826236447
7	-22	22	20,91650066	-	1,051801176
8	-28	28	22,36067977	-	1,252198067
9	-34	34	23,71708245	-	1,433565873
10	-40	40	25	-	1,6

Funkcijų *ERFC(x)* bei *ERF(x)* išraiškos taip pat skaičiuojamos skaičiuoklės pagalba. Šių funkcijų apskaičiuotos reikšmės dešimčia iteracijų pateiktos žemiau esančioje lentelėje:

Iteracijos numeris	$ERFC(x)$	$ERF(x)$	$D_N(M)$
1	0,012266	0,987734	0,006133031
2	0,311572	0,688428	0,155786163
3	0,836354	0,163646	0,418177098
4	0,720515	0,279485	0,639742606
5	0,423711	0,576289	0,788144601
6	0,242615	0,757385	0,878692278
7	0,13689	0,86311	0,931554817
8	0,076581	0,923419	0,961709288
9	0,042625	0,957375	0,978687617
10	0,023652	0,976348	0,98817419

Turint visus reikiamus duomenis nubraižoma diagrama:



Gauti duomenis rodo, kad tikimybė pabaigti projektą jau po septynių iteracijų yra gana didelė ir su kiekviena sekančia iteracija ši tikimybė vis didėja.

Aprašyto metodo pagalba galima sudaryti praktiškai kiekvieno projekto užbaigimo laikotarpio prognozes.

REZULTATAI IR IŠVADOS

Magistro baigiamajame darbe „Laiko įverčių analizė ir vizualizavimas tiriamąjį darbą organizuojant remiantis „Agile“ praktikomis“ yra išanalizuota darbo organizavimo procesų ir projektų valdymo metodų mokslinė ir metodinė literatūra, palyginti dažniausiai naudojami darbo organizavimo metodai su klasikiniais metodais, ištirti bei susisteminti darbo organizavimo procesų ir projektų valdymo metodai.

Darbe pateiktas darbo proceso užbaigimo laikotarpio prognozių sudarymo metodas, leidžiantis apskaičiuoti projekto užbaigimo laikotarpį. Taip pat pateiktas aprašyto metodo praktinis pavyzdys su apskaičiuotų reikšmių lentelėmis bei diagramomis.

Darbe pateiktas pagerintas darbo proceso vaizdavimo metodas, leidžiantis įvertinti ir aiškiau pavaizduoti besikeičiančią darbų apimtį tarp iteracijų ir iteracijų metu. Iki šiol šaltiniuose buvo pateikiami metodai, nagrinėjantys tik statinės darbų apimties vaizdavimą.

Šis darbas galėtų būti naudingas daugelyje veiklos sričių. Pasirinkus dėstytojo specialybę, įgytas žinias galima pritaikyti laboratorinių, praktinių bei kursinių darbų organizavimo procese. Šios žinios galėtų būti naudingos visur, kur tik gali prireikti darbo organizavimo žinių bei įgūdžių.

SUMMARY

Analysis of time estimates and visualization organizing reseach according to Agile practices

Student: Jelena Zubova

Supervisor: Dr. Rimgaudas Laucius

In the Master Thesis "Analysis of time estimates and visualization organizing research according to Agile practices" the following tasks are accomplished: scientific and methodical literature related to the process of work organizing and the methods of project management is reviewed; the commonly used methods of work organizing are compared to the classic methods; the methods of work organizing and project management are analyzed and systematized.

The method of forecasting the time of working process is presented in the Thesis. The method allows calculating the time of working process. The example of application of this method is presented as well including tables with calculated values and diagrams. The improved method of working process visualization is presented in the Thesis.

The method allows evaluating and depicting the changing volume of work between iterations as well as during iterations in a more readable way. So far only the methods dealing with the visualization of static volume of work have been proposed in the literature.

The Thesis could be useful in many fields. In regard to teaching, the acquired knowledge could be used when organizing laboratory works, practicals and project works. The knowledge could be useful in every area that requires the skills of work organizing.

LITERATŪROS SĄRAŠAS

1. [AG01] *Manifesto for Agile Software Development* (2001). http://agilemanifesto.org/, [žiūrėta 2012-03-04].
2. [AP01] *Principles behind the Agile Manifesto* (2001). http://agilemanifesto.org/principles.html, [žiūrėta 2012-03-04].
3. [BK07] V. Bagdonavičius, J. Kruiopis. *Matematinė statistika, I dalis* (2007).
4. [BK09] G. Bortkevič-Krunglevičienė. *Projektų vadybos metodikų pasirinkimo ir pritaikymo būdai informacinių technologijų projektuose* (2009).
5. [Bo11] Б. Вольфсон. *Гибкие методологии разработки* (2011).
6. [BPĮ07] Baltijos Programinė Įranga. *Programinės įrangos kūrimo procesai* (2007).
7. [CG12] A. Cabri, M. Griffiths. *Earned Value and Agile Reporting.*
8. [CPVA10] CPVA2010. *Projekto priežiūra.* (2007-2013).
9. [ČM06] V. Čekanavičius, G. Murauskas. *Statistika ir jos taikymai* (2006).
10. [De12] Dekona (2012). *Lean Manufacturing.* http://www.dekona.lt/index.php?page_id=160, [žiūrėta 2012-03-07].
11. [EM] Eduardo Miranda. *Agile Monitoring using the Line of Balance* (2009).
12. [ID12] Verslo konsultacijos Imperatum (2007-2012). *Apie Lean.* http://www.imperatum.lt/IMPERATUM/Lean_sistema.html, [žiūrėta 2012-03-08].
13. [In12] Informicus (2006-2012). *Extreme Programming*. http://www.informicus.ru/default.aspx?SECTION=6&id=95, [žiūrėta 2012-03-06].
14. [IS11] Verslo konsultacijos Imperatum (2007-2011). *Apie Lean.* http://www.imperatum.lt/IMPERATUM/Lean_sistema.html, [žiūrėta 2012-03-07].
15. [Pa12] D. Palivonienė. *Projektų valdymas: kaip pasiruošti sėkmei?*
16. [SD12] Scrum Trek (2008-2012). *Daily Scrum.* http://scrumtrek.ru/scrum/Daily-scrum/, [žiūrėta 2012-03-05].
17. [SS12] Scrum Trek (2008-2012). *Sprint review meeting.* http://scrumtrek.ru/scrum/Sprint-review-meeting/, [žiūrėta 2012-03-05].
18. [SŠ04] I. Simanavičiūtė, V. Šilingienė (2004). *Darbo organizavimo esmė ir turinys.* http://www.lzuu.lt/jaunasis_mokslininkas/smk_2004/Pletra/Simanaviciute_Ingrida.htm, [žiūrėta 2012-03-06].
19. [Su12] Sukhoi (2004-2012). *Карта потока создания ценностей.* http://sukhoi.org/dku/item1/article2/, [žiūrėta 2012-03-08]. [Tu08] L. Tutkutė. *MDA ir programų kodo generavimas* (2008).

20. [Šk09] J. Škatulaitė. *Aplikacijų reikalavimų valdymo tyrimas* (2009). [Wi12] Wikipedia (2012). *Экстримальное программирование.* http://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D1%81%D1%82%D1%80%D0%B5%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5, [žiūrėta 2012-03-06].
21. [Ze12] Zenegment. *Freakonomics of Scrum: Детали.* http://cartmendum.livejournal.com/10339.html, [žiūrėta 2012-04-16].
22. [Ше11] Е. Шеретов (2011). *Управление Дедлайном.* http://www.slideshare.net/sheretovev/deadline-management, [žiūrėta 2012-03-12].
23. [Ха12] Хантим (2006-2012). *Kanban Development*. http://habrahabr.ru/post/64997/, [žiūrėta 2012-03-07].