UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

MAGISTRSKO DELO

SPLETNO ORODJE ZA SVETOVANJE PRI SCRUM-U

MATEJ FILIPOVIĆ

2021
MAGISTRSKO DELO
MATEJ FILIPOVIĆ

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN INFORMACIJSKE TEHNOLOGIJE

Magistrsko delo

Spletno orodje za svetovanje pri Scrum-u

(Web tool for Scrum consulting)

Ime in priimek: Matej Filipović
Študijski program: Računalništvo in informatika, 2. stopnja
Mentor: doc. dr. Peter Rogelj
Somentor: doc. dr. Branko Kavšek

Koper, junij 2021

Ključna dokumentacijska informacija

Ime in PRIIMEK: Matej FILIPOVIĆ

Naslov magistrskega dela: Spletno orodje za svetovanje pri Scrum-u

Kraj: Koper

Leto: 2021

Število listov: 89 Število slik: 15 Število tabel: 40

Število referenc: 56

Mentor: doc. dr. Peter Rogelj

Somentor: doc. dr. Branko Kavšek

UDK: 004.774(043.2)

Ključne besede: Scrum, ScrumBut, vpeljava Scrum-a, težave po uvedbi Scrum-a, nasveti za reševanje težav.

Izvleček:

Ogrodje Scrum pridobiva na priljubljenosti in je vse pogosteje uporabljeno v različnih podjetjih. Pri vpeljavi in uporabi Scrum-a podjetja pogosto ne upoštevajo vseh potrebnih dejavnikov in ga ne uporabljajo na najučinkovitejši način. V magistrskem delu smo predstavili osnovno strukturo Scrum-a (vloge, sestanke in potek dela) ter posledice, zakaj se Scrum ne uporablja učinkovito in kaj lahko izboljšamo. Bistvene podatke, namenjene izboljšanju uporabe Scrum-a, smo predstavili v spletni strani. Namen spletne strani je povečanje dostopnosti do pridobljenih podatkov v raziskavi magistrskega dela. Na ta način želimo Scrum skupnosti pomagati priti do nasvetov, ki jim lahko pomagajo izboljšati doseganje boljše poslovne učinkovitosti in boljšega počutja zaposlenih.

Key document information

Name and SURNAME: Matej FILIPOVIĆ

Title of the thesis: Web tool for Scrum consulting

Place: Koper

Year: 2021

Number of pages: 89 Number of figures: 15 Number of tables:40

Number of references: 56

Mentor: Peter Rogelj, PhD

Co-Mentor: Branko Kavšek, PhD

UDC: 004.774(043.2)

Keywords: Scrum, ScrumBut, Scrum integration, Difficulties after integrating Scrum, Suggestions for problem solving.

Abstract:

The Scrum framework is gaining popularity and it is more often used in various companies. When implementing and using Scrum, companies often do not take into account all the necessary factors to use Scrum in the most efficient way. In the master´s thesis, we presented the basic structure of Scrum (roles, meetings and workflow) and presented the consequences of why Scrum is not used effectively and what can be improved. We presented essential data intended to improve Scrum usage on a website. The purpose of website is to increase the accessibility of the data obtained in the master thesis´s research. In this way, we want to help Scrum community with tips that can help them improve business efficiency and employee well-being.

KAZALO VSEBINE

KAZALO PREGLEDNIC

KAZALO SLIK IN GRAFIKONOV

ZAHVALA

Rad bi se zahvalil svoji družini in ženi za moralno podporo pri pisanju magistrskega dela. Zahvalil bi se tudi mentorju doc. dr. Peteru Rogelju za hitro odzivnost in nasvete.

1 UVOD

Scrum je ogrodje, ki me spremlja že od samega začetka moje poklicne kariere. Uporaba Scrum-a se med različnimi podjetji zelo razlikuje, saj si podjetja prilagodijo metode v ogrodju svojim potrebam. Na podlagi mojih izkušenj podjetja prilagajajo Scrum z nepopolnim razumevanjem vseh posledic, ki so lahko tudi zelo negativne. Takšen način uporabe Scrum-a lahko vodi do zaključka, da ni učinkovit in da ga ni smiselno uporabljati. Raznolikost in nepravilen način uporabe Scrum-a sta me motivirala, da poskusim opisati najbolj učinkovit način uporabe le-tega.

Cilj magistrske naloge je raziskati in opisati pozitivne in negativne vidike uporabe Scrum ogrodja in pomagati podjetjem, da ugotovijo, kako Scrum najbolj učinkovito uporabljati. Z nasveti, kaj morajo spremeniti, da lahko pridejo do bolj učinkovite strukture in s tem do povečane poslovne učinkovitosti. Slednje bo mogoče izvesti z uporabo spletne strani, ki je narejena v sklopu magistrske naloge.

2 OPIS SCRUM-A

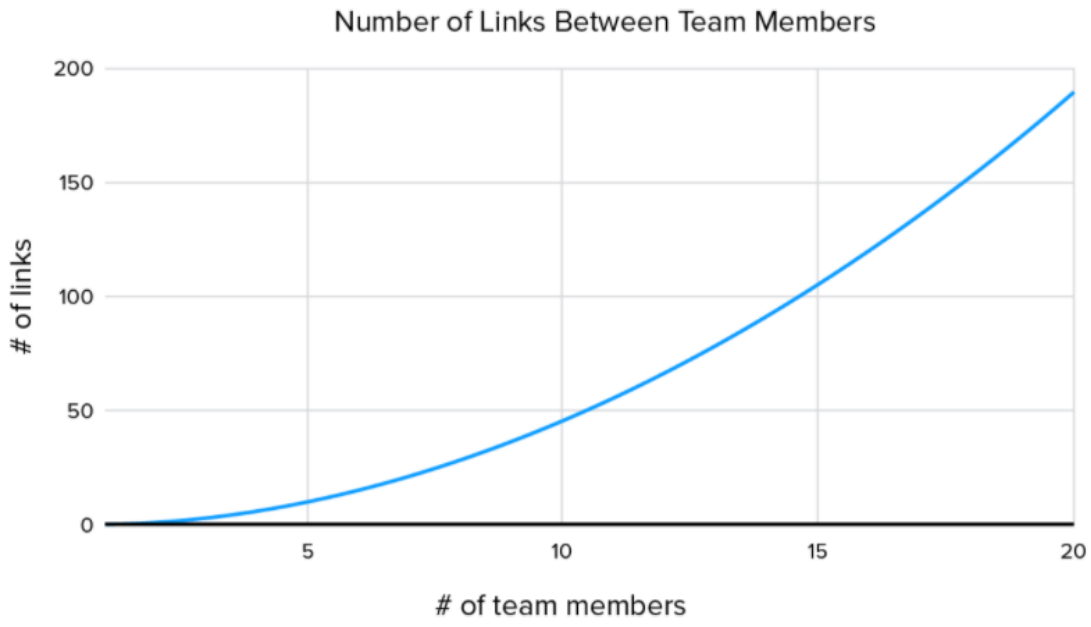
Scrum je ena izmed najbolj znanih agilnih ogrodij. Izumila sta jo Ken Schwaber in Jeff Sutherland leta 1993. Scrum je bil prvič omenjen v članku »Agile development: Lessons Learned from the First Scrum« [47].

Osnova Scrum ogrodja je iterativni razvoj z razvrščanjem zahtev, kar nam dovoljuje, da razvijemo najbolj uporabne in pomembne funkcionalnosti zgodaj, že po prvih nekaj iteracijah. Daje nam popoln nadzor nad smerjo in velikostjo projekta, saj se lahko po koncu vsake iteracije odločimo, ali bomo dodali nove funkcionalnosti, jih odstranili ali spremeniti obstoječe. Iteracije omogočajo tudi, da hitreje ugotovimo, kadar produkt ne prinese zadosti dodane vrednosti. Scrum ogrodje omogoča dobro preglednost nad projektom z uporabo različnih orodij, kot so: padajoči diagram spremljanja napredka (ang. burn-down chart), seznam uporabniških zgodb (ang. product backlog) in seznam nalog iteracije (ang. sprint backlog). K večji preglednosti nam pomagajo tudi sestanki iz Scrum-a: sestanek za načrtovanje iteracije (ang. sprint planning), dnevni sestanki, sestanek za predstavitev rezultatov iteracije (ang. sprint review) in retrospektivni sestanek iteracije (ang. sprint retrospective). Dobra preglednost nad projektom omogoča več nadzora nad potekom projekta kot tudi hitrejše odkrivanje težav, ki lahko nastanejo v projektih [3].

Scrum deluje najbolje v manjših ekipah, ki jih sestavlja od pet do devet razvijalcev. Manjše število članov ekipe je lahko slabost, saj nam lahko oteži dokončanje projekta zaradi

premajhne raznolikosti članov in s tem posledično manj ljudi z različnimi izkušnjami. Uporaba Scrum metodologije v velikih ekipah nam lahko podaljša sestanke. Dnevni sestanki, sestanek za načrtovanje iteracije in ostali sestanki se podaljšajo do te mere, da na projektu več ne delamo učinkovito [42].

Slika 1 prikazuje, kako se število povezav med člani Scrum ekipe povečuje s povečanjem števila članov ekipe. Pri treh osebah obstaja 36 povezav, medtem ko imamo pri 15 osebah že 100 povezav med člani ekipe. Ekipa s 15 člani lahko učinkovito deluje le v primeru, če bi bile naloge zelo dobro opredeljene in bi se le redko medsebojno prekrivale. Le-to pa je zelo težko izvedljivo pri razvoju produkta [9].



Slika 1: Število povezav med člani ekipe [9]

Pravilna izvedba Scrum-a je lahko zelo težavna, saj Scrum, tako kot druga agilna ogrodja, močno temelji na timskem delu, komunikaciji, zaupanju in na prenosu odgovornosti in avtoritete. Vse to predstavlja veliko kulturno spremembo, ki je še posebej vidna pri podjetjih, kjer so v preteklosti uporabljali tradicionalne metode dela. Takšna kulturna sprememba zahteva čas in trdo delo, da je uspešno sprejeta s strani zaposlenih in vodstva [3].

Agilne metode poudarjajo dva koncepta: prikaz delujoče aplikacije in učinkovitost ljudi, ki medsebojno dobro sodelujejo. Delujoča aplikacija omogoča razvijalcem in stranki, da preizkusijo novo razvite funkcionalnosti in tako bolje ocenijo, kako nadaljevati razvoj. S tem razvijalci in stranka hitreje najdejo skupno razumevanje, kako bi moral končni produkt izgledati in delovati. V Scrum-u se večinoma člani ekipe s stranko pogovarjajo iz oči v oči,

saj je to veliko bolj učinkovito kot pisanje in branje dokumentov, kar velikokrat velja pri podjetjih, ki uporabljajo tradicionalne metode dela. Ko se razvijalci pogovarjajo s strankami in ostalimi interesnimi skupinami, lahko hitreje ugotovijo najbolj bistvene težave, prilagodijo prioriteto uporabniških zgodb in pregledajo druge možne rešitve. Le tako s timskim delom dosežemo najvišjo učinkovitost [29].

Scrum vzpodbuja komunikacijo in timsko delo znotraj ekipe z nenehnimi in načrtovanimi sestanki, kjer ima vsak sestanek svoj namen in okviren načrt izvedbe [13], [22]. Williams [55] je pokazal, da tudi po desetletju uporabe agilnih praks strokovnjaki vedno poudarjajo pomen komunikacije in sodelovanja pri razvoju programske opreme. Le-ta je eden od glavnih razlogov za uspešno izvedbo agilnih projektov.

2.1 VLOGE V SCRUM-U

Scrum ekipa je sestavljena iz treh vlog, ki se med seboj razlikujejo, toda hkrati so zelo pomembne pri razvoju projekta in učinkoviti uporabi Scrum-a. Vloge so: lastnik izdelka (ang. product owner), skrbnik Scrum projekta (ang. scrum master) in član razvojne ekipe. V nadaljevanju so posamezne vloge podrobneje opisane.

2.1.1 Lastnik izdelka

Vloga lastnika izdelka je ena najpomembnejših vlog v Scrum ekipi in pogosto tudi najtežja. Odgovoren je za opredelitev zahtev za razvojno ekipo in da projekt doseže zastavljene cilje. Lastnik izdelka je povezan z vsemi osebami, ki so vključene v projekt. Glede na vse sprejete informacije mora najti ravnovesje med različnimi lastnostmi izdelka, tako da bo končni izdelek najbolje ustrezal željam stranke. Lastnik izdelka skrbi za vzdrževanje in razvrščanje uporabniških zgodb po prednosti v seznamu uporabniških zgodb. Le-ta je zbirka vseh uporabniških zgodb, za katere se je lastnik izdelka z razvojno ekipo odločil, da jih bodo razvili v projektu. Seznam uporabniških zgodb se spreminja in razvija v življenjskem ciklu projekta. Tehnične težave, spremenjene želje končne stranke ali kakšni drugi vplivi lahko povzročajo, da se mora seznam uporabniških zgodb prilagoditi. V takšni situaciji mora lastnik izdelka pridobiti mnenja od ostalih udeležencev v projektu in se po svoji najboljši presoji odločiti, kaj je potrebno v projektu prilagoditi [52].

Ena od najpomembnejših nalog lastnika izdelka je, da se odloči, kaj ni potrebno upoštevati pri končnem produktu in prevzeti posledice te odločitve. Nujno je, da v sodelovanju s stranko in razvojno ekipo zavrne nove zahteve, ki niso potrebne [49].

Lastnik izdelka mora imeti dobre komunikacijske sposobnosti in biti dober v pogajanjih. Za vodenje razvoja in usklajevanje želja zainteresiranih strani mora imeti dovolj avtoritete in podpore izvršnega vodstva [52].

Vloga lastnika izdelka se odlično dopolnjuje z vlogo skrbnika Scrum projekta, saj je prvi odgovoren, "kaj" storiti, skrbnik Scrum projekta pa "kako" to storiti [30].

2.1.2 Skrbnik Scrum projekta

Odgovornost skrbnika Scrum projekta je, da se proces Scrum pravilno izvaja. Le-ta nadzoruje komunikacijo v ekipi in poskrbi, da se vse morebitne ovire odstranijo. Njegova odgovornost je, da odstrani vse ovire, ki bi onemogočile, da ekipa doseže cilje iteracije ali celotnega projekta. Poleg tega skrbnik Scrum projekta rešuje nesoglasja v Scrum ekipi ter utira pot projektu v podjetju in tako zagotavlja, da spremembe znotraj podjetja ne zmanjšajo produktivnosti ekipe [41].

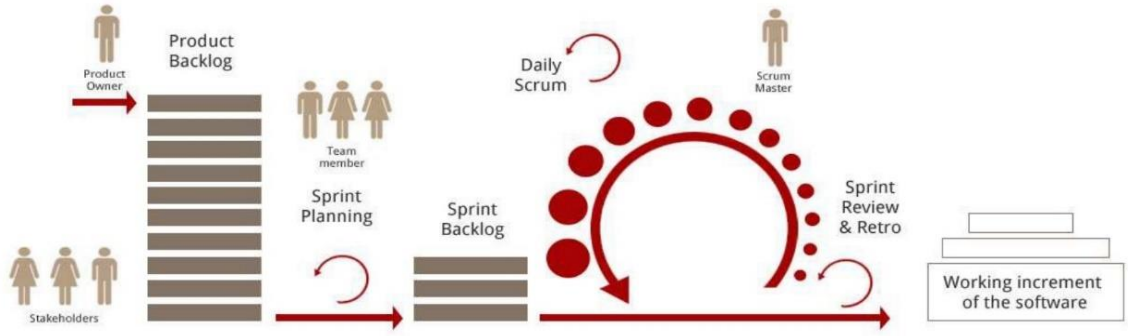
2.1.3 Razvojna ekipa

Razvojno ekipo lahko opišemo kot majhno, medfunkcijsko (ang. cross-functional) skupino ljudi, katere člani imajo znanje, da razvijejo končni produkt. Le-ta deluje samostojno, edine točke, kjer je potrebno več koordinacije z lastnikom izdelka in skrbnikom Scrum projekta, so sestanki iz Scrum okolja, ki so: dnevni sestanek, sestanek za načrtovanje iteracije, sestanek za predstavitev rezultatov iteracije in retrospektivni sestanek iteracije. Le-ti so podrobneje predstavljeni v nadaljevanju.

2.2 SCRUM CIKEL

Vse prej opisane vloge in sestanki morajo imeti jasno strukturo, kdaj se posamezen sestanek zgodi in kdo je zanj odgovoren. Za to skrbi Scrum cikel, ki ima enako strukturo ne glede na dolžino iteracije. Slika 2 spodaj prikazuje podrobnejši potek Scrum cikla, ki je sestavljen iz petih ključnih korakov:

1. definiranje ali posodabljanje seznama uporabniških zgodb z razvojno ekipo;
2. sestanek za načrtovanje iteracije;
3. začetek in potek iteracije:
    - razvoj uporabniških zgodb iz seznama nalog iteracije,
    - dnevni sestanki;
4. konec iteracije:
    - sestanek za predstavitev rezultatov iteracije,
    - retrospektivni sestanek iteracije,
5. cikel se ponovi s točko 1, dokler projekt ni končan.



Slika 2: Scrum cikel z akterji in orodji [50]

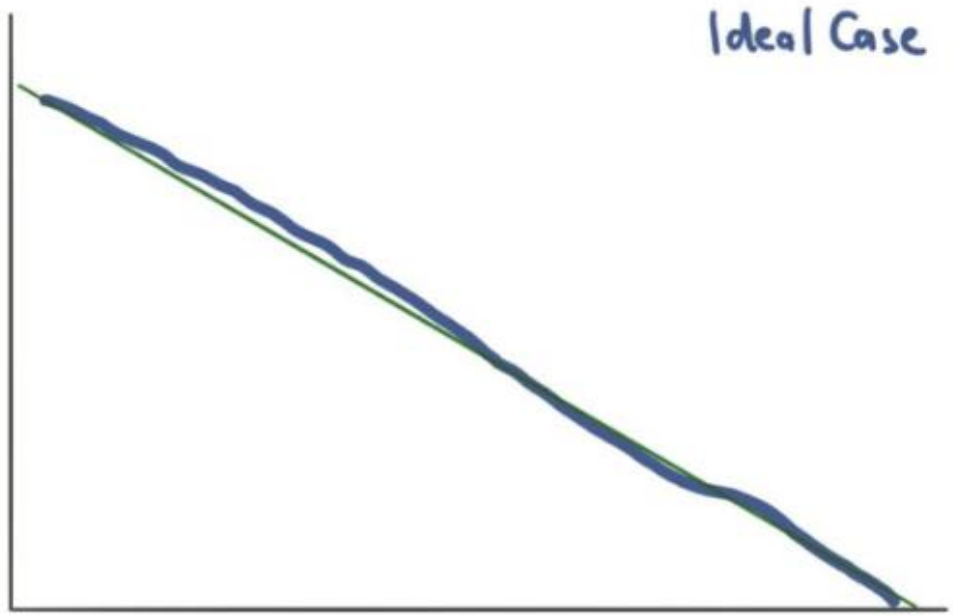
2.2.1 Table z nalogami in padajoči diagram spremljanja napredka

Pri izvedbi Scrum cikla sta pogosto uporabljeni metodi tabla z nalogami (ang. task board) in padajoči diagram spremljanja napredka. Tabla z nalogami je fizična tabla, ki se po navadi nahaja v pisarni razvojne ekipe in je uporabljena za jasen pregled nad potekom uporabniških zgodb v iteraciji. Razvijalci jo uporabljajo (večinoma med dnevnim sestankom) za:

- dodajanje in odstranjevanje manjših/tekočih nalog,
- prikaz in posodabljanje padajočega diagrama spremljanja napredka,
- vizualno predstavitev, kateri razvijalec dela na kateri uporabniški zgodbi/nalogi,
- druge aktivnosti, ki pomagajo pri jasnosti in preglednosti iteracije ali projekta (npr. predstavitev različnih metrik z diagrami, predstavitev vmesnih ciljev projekta …).

Padajoči diagram spremljanja napredka je ena od najbolj uporabljenih metod v Scrum-u za vizualno predstavitev, koliko dela je še preostalo v iteraciji ali v celotnem projektu. Pogosto imamo na tabli z nalogami padajoči diagram spremljanja napredka za trenutno iteracijo, katerega pogosto ročno dopolnjujemo. Na navpični premici imamo podatek, koliko dela je bilo načrtovanega za iteracijo, izraženo v točkah (ang. story points). Na vodoravni premici pa imamo trajanje iteracije po dnevih. Na sestanku za načrtovanje

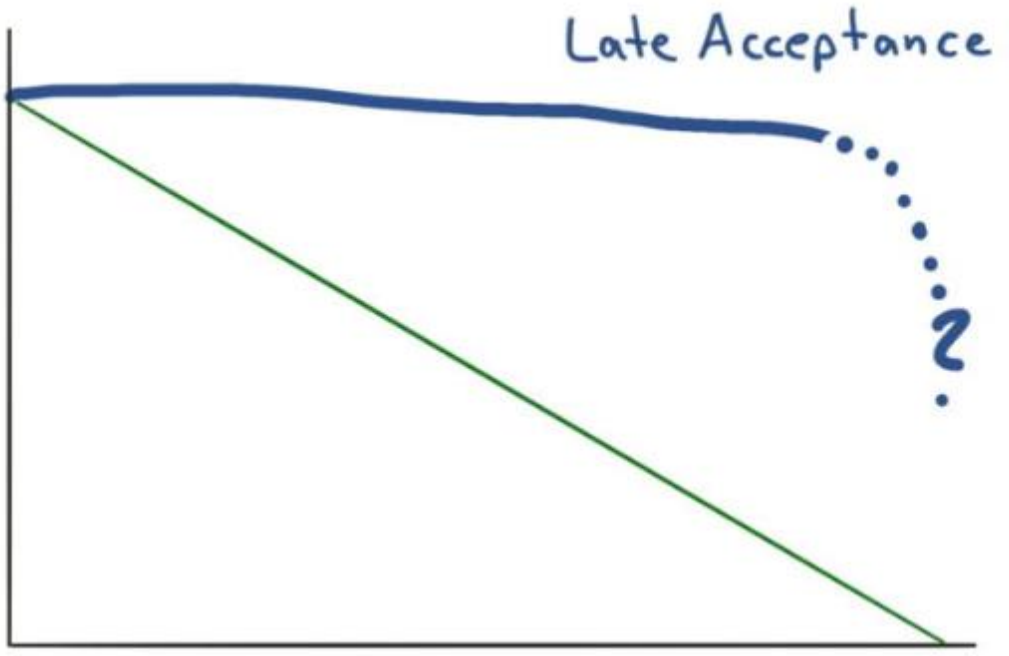
iteracije se razvojna ekipa sama odloči, koliko dela/uporabniških zgodb bo opravila v iteraciji. Na podlagi količine dela in določenega časa trajanja iteracije oblikujemo padajoči diagram spremljanja napredka za novo iteracijo. Le-ta je pogosto uporabljen pri Scrum-u, saj omogoči hitro odkrivanje raznolikih težav pri načrtovanju in poteku dela. Tako lahko na primer hitro vidimo, če iteracija poteka, kot je bila načrtovana. Kadar iteracija poteka po načrtu, je stopničasta črta v bližini diagonale, kot je prikazano na sliki 3.



Slika 3: Padajoči diagram spremljanja napredka – Idealni potek iteracije [54]

Z vizualnim prikazom, ki nam omogoča padajoči diagram spremljanja napredka, lahko hitro vidimo tudi druge vzorce, ki motijo normalen potek iteracije. Zaradi tega lahko hitreje odreagiramo in tako vrnemo potek iteracije na pravo pot. Če v trenutni iteraciji ne moremo pravočasno odreagirati, smo dobili nove informacije, ki jih bomo upoštevali v naslednji iteraciji. V nadaljevanju so našteti in grafično prikazani vzorci, ki pomagajo odkriti moten potek iteracije:

- Pozno sprejemanje uporabniških zgodb.

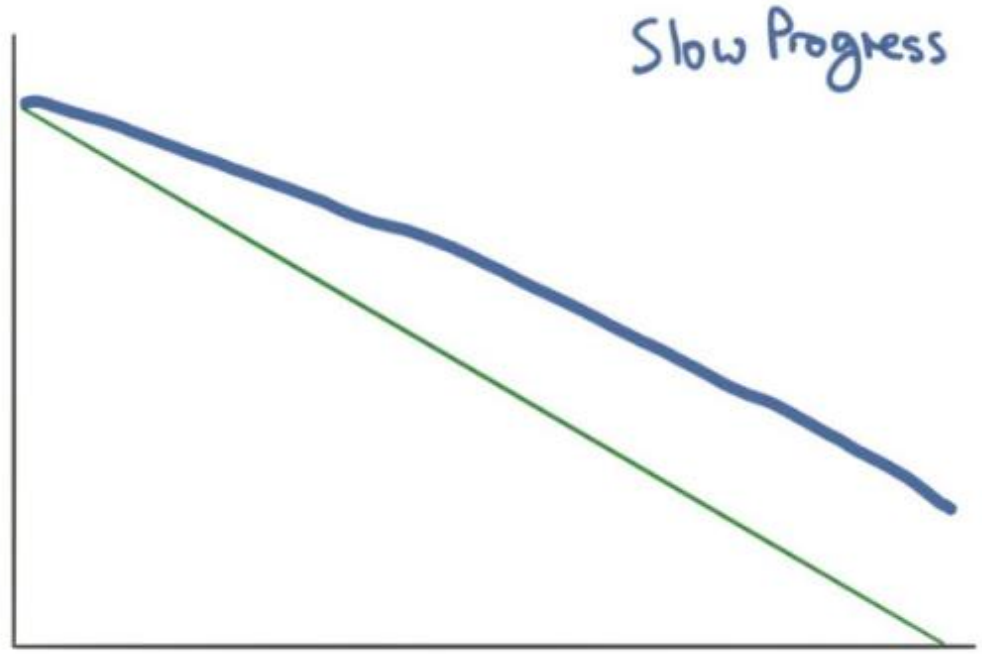


Slika 4: Padajoči diagram spremljanja napredka – Pozno sprejemanje uporabniških zgodb [54]

Za uspešno izpeljavo iteracije je potrebno uporabniške zgodbe sprejemati postopoma skozi iteracijo. Na ta način zagotovimo zadosti časa, da lastnik izdelka poda povratne informacije razvojni ekipi ter da razvijalci posodobijo prvotno funkcionalnost. Potrebno je načrtovati tudi čas za preverjanje definicije končanosti (ang. Definition of Done), ki skrbi za določen standard kakovosti opravljenega dela na uporabniških zgodbah, ter za test funkcionalnosti. Vse zgoraj opisane naloge je zelo težko opraviti, če so vse uporabniške zgodbe sprejete na koncu iteracije. V praksi to pogosto pomeni, da ekipa ne razvije vseh začrtanih uporabniških zgodb v iteraciji ter s tem ne doseže cilja iteracije. Med drugim pozno sprejemanje uporabniških zgodb povzroči veliko stresa razvojni ekipi. Pozno sprejemanje uporabniških zgodb se lahko zgodi v sledečih primerih:

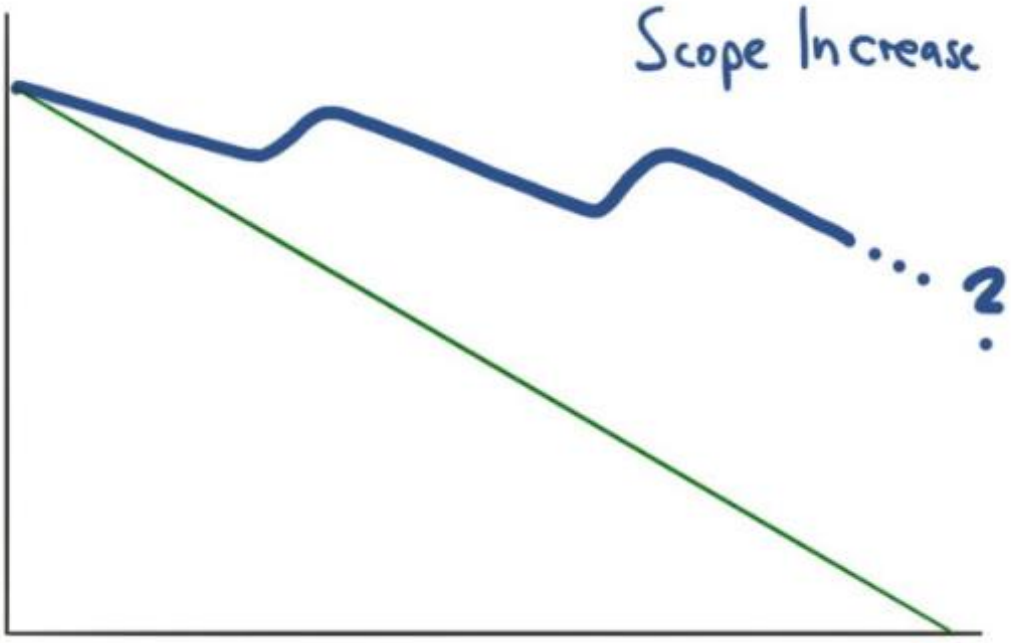
- lastnik izdelka ni prisoten in ne more pravočasno podati povratne informacije razvojni ekipi,
- razvojna ekipa narobe oceni zahtevnosti uporabniških zgodb,
- razvojna ekipa v seznam nalog iteracije izbere samo zahtevne uporabniške zgodbe, za katere potrebujejo več razvojnega časa [54].

- Počasen napredek



Slika 5: Padajoči diagram spremljanja napredka - Počasen napredek [54]

- Povečanje obsega med iteracijo

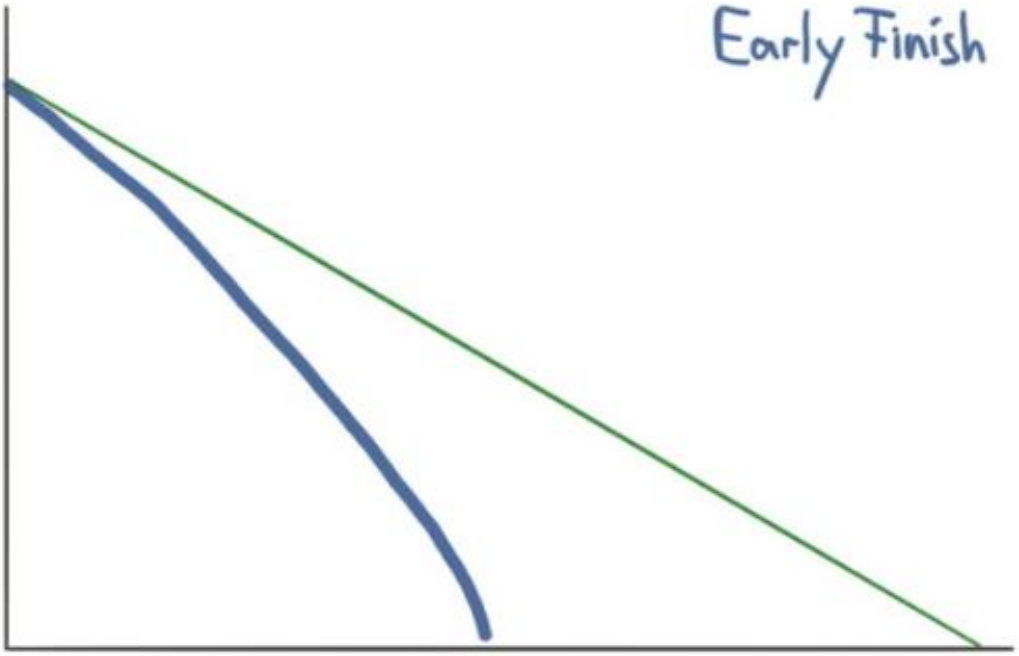


Slika 6: Padajoči diagram spremljanja napredka – Povečanje obsega med iteracijo [54]

Glavni razlogi za povečanje obsega med iteracijo so:

- vodstvo ali stranka zahteva spremembo seznama nalog iteracije med potekom iteracije,
- kritična napaka v produkciji zahteva takojšnjo razrešitev za normalno delovanje produkta [54].

- Zgoden zaključek



Slika 7: Padajoči diagram spremljanja napredka – Zgoden zaključek [54]

Zgodnji zaključek se pogosto zgodi v primerih, ko:

- razvojna ekipa preceni zahtevnost uporabniških zgodb v iteraciji.
  - To lahko nakazuje, da vodstvo uporablja število opravljenih točk v iteraciji kot pomembno metriko, s katero nadzoruje uspešnost razvojne ekipe. V tem primeru je podjetje usmerjeno k rezultatom in ne sprejema možnosti za neuspeh. Takšen nadzor postavlja napačne spodbude za uspešen razvoj produkta.
- Razvojna ekipa želi zagotoviti dodaten čas za reševanje tehničnih težav, za katere ne dobijo namenjenega časa.
  - To lahko nakazuje, da trenutna dodelitev virov zanemarja dolgoročno kakovost produkta. V tem primeru je potrebno paziti, da rezultat trenutne iteracije ni pomembnejši od dolgoročnega rezultata [54].

2.2.2 Dolžina iteracije

Predlagana dolžina iteracije v Scrum-u je med enim in štirimi tedni. Dolžino trajanja iteracije mora določiti ekipa sama brez lastnika izdelka, interesnih skupin ali vodstva. Pri neizkušenih ekipah ekipi pomaga dolžino določiti skrbnik Scrum projekta. Dolžina se določi glede na to, kakšen produkt razvijamo v projektu in kako dolgo traja projekt. Na primer pri krajših projektih imamo več koristi s krajšimi iteracijami, saj pridobimo več povratnih informacij in tako bolje zagotovimo uspešnost projekta.

Prednosti krajše iteracije:

- Pogostejši retrospektivni sestanki iteracije in sestanki za predstavitev rezultatov iteracije povzročijo hitrejše povratne informacije in s tem hitrejše prilagoditve na produktu in razvojnem procesu. Le-to je predvsem zaželeno pri razvoju produktov, ki vsebujejo večjo tehnično zahtevnost in zaradi tega več tveganj. Večja negotovost pri takšnem razvoju se tako zmanjša s pogostejšimi Scrum sestanki.
- Krajši sestanki za načrtovanje iteracije zmanjšajo možnost, da izberemo neustrezno definirane naloge globoko iz seznama uporabniških zgodb. Neprimerno definiranim uporabniškim zgodbam drugače tudi rečemo, da ne dosegajo definicije pripravljenosti (ang. Definition of Ready).

Slabosti krajše iteracije:

- Več časa porabimo za pripravo na iteracijo in za Scrum procese v iteraciji v primerjavi z daljšimi iteracijami. S tem se razmerje časa za dejanski razvoj skrajša.
- Zaradi kratke iteracije preizkuševalci nimajo časa za pripravo popolnoma avtomatičnega testiranja produkta. Le-to lahko pomeni, da bo preizkuševalec moral opraviti več ročnega testiranja ter zaradi tega mogoče ne bo mogel slediti napredku razvojne ekipe. Tako je testno voden razvoj (ang. TTD – Test Driven Development) težko izpeljiv in veliko uporabniških zgodb ne ustreza definiciji končanosti na koncu iteracije.

Scrum ekipe se lahko tudi soočajo z veliko motnjami in nujnimi spremembami med trajanjem iteracije (npr. zaradi stranke, ki zahteva hitre prilagoditve). V takšnem primeru predlagamo, da se dolžina iteracije zmanjša do te mere, da je ekipa lahko odzivna na spremembe stranke, brez spreminjanja nalog v seznamu nalog iteracije. V ekstremnih primerih določene ekipe delujejo v okolju, kjer morajo biti spremembe takojšnje (npr. produkcijsko okolje, kjer vsaka sekunda šteje). Takšne ekipe pogosto razmišljajo o uporabi iteracij, ki so krajše od tedna, kar ni priporočljivo (glej prej opisane slabosti krajše iteracije). V takšnem primeru bi svetovali uporabo drugačnih agilnih metodologij, kot je na primer Kanban. Po drugi strani se ekipe, ki uporabljajo iteracije daljše od štirih tednov,

lahko soočajo s težavami tradicionalnih pristopov, kot je slapovni (ang. watefall), in ne bodo izkoristile prednosti Scrum ogrodja.

Dodatne težave lahko nastanejo, če Scrum ekipa pogosto spreminja dolžino iteracije. To je sprejemljivo le na začetku projekta pri novo sestavljeni Scrum ekipi, ko ekipa ugotavlja, katera dolžina je najbolj primerna za njih. Nekatere potencialne težave s pogosto menjavo dolžine iteracije so:

- načrtovanje iteracije postane težje, saj razvijalci niso prepričani, koliko dela lahko opravijo v naslednji iteraciji;
- spreminjajoči ritem povzroča, da člani Scrum ekipe in preostale interesne skupine (produktni manager, stranka, sistemski arhitekt …) niso prepričani, kdaj sledijo naslednji Scrum dogodki in zato težje načrtujejo svoj čas. To otežuje prisotnost vseh potrebnih udeležencev na sestankih. V primeru iteracij s fiksno dolžino je leto lažje, saj vsi udeleženci natančno vedo, kdaj se bo zgodil naslednji Scrum dogodek.

2.2.3 Ocenjevanje velikosti uporabniških zgodb

Pri temi ocenjevanja velikosti uporabniških zgodb v seznamu uporabniških zgodb pogosto opazimo dva vzorca: »ocenjevanje s točkami« in »ocenjevanje z urami«. Bolj uporabljen pristop pri Scrum-u je, da ekipe uporabljajo točkovno oceno za ocenjevanje količine dela na uporabniških zgodbah namesto ocen z urami. Scrum ekipe za točkovno oceno večinoma uporabljajo številke Fibonaccijevega zaporedja (1, 2, 3, 5, 8 …). Za vsako številko opredelijo referenčno razvojno nalogo, kjer 1 predstavlja enostavno nalogo (npr. na spletno stran dodaj gumb z enostavno funkcijo), večje številke pa posledično vse bolj zahtevne naloge. Točkovna ocena v primerjavi z oceno z urami:

- zmanjša pritisk na razvijalce v primeru, ko naloga ni pravočasno zaključena;
- omogoča lažje skupinsko ocenjevanje zahtevnosti uporabniške zgodbe (npr. z metodo načrtovalni poker);
- zmanjša čas, ki ga Scrum ekipa porabi za ocenjevanje velikosti. Ocenjevanje z urami je bolj podrobno in vzame več časa. Ljudje po navadi ne ocenimo količine potrebnega dela dovolj natančno, zato ta bolj podrobna ocena ne prinese dodane vrednosti ekipi.

2.3 SCRUM SESTANKI

V nadaljevanju so predstavljeni Scrum sestanki, ki so ključni za uspešen potek Scrum-a.

2.3.1 Sestanek za načrtovanje iteracije (ang. Sprint planning)

Na začetku vsake iteracije poteka sestanek za načrtovanje iteracije. Sestanka se udeležijo lastnik izdelka, skrbnik Scrum projekta, člani razvojne ekipe in v določenih primerih tudi stranka. Cilj sestanka je priti do skupnega dogovora med lastnikom izdelka in razvojno ekipo, kaj bodo razvili v naslednji iteraciji. Sestanek je razdeljen na dva različna dela. V prvem delu sestanka lastnik izdelka in razvojna ekipa (s pomočjo skrbnika Scrum projekta) pregledajo prednostne uporabniške zgodbe iz seznama uporabniških zgodb. Prednostne uporabniške zgodbe določi lastnik izdelka pred sestankom, sama prioriteta pa se lahko tudi spremeni med sestankom zaradi dodatnih informacij s strani razvijalcev. Lastnik izdelka in razvojna ekipa skupaj razpravljajo o ciljih in vsebini prednostnih uporabniških zgodb. Razvojna ekipa mora dobro razumeti razmišljanje lastnika izdelka, da bi razvila izdelek po želji stranke. Po tem, ko so razvojni ekipi razumljivi cilji posameznih uporabniških zgodb iz seznama uporabniških zgodb, se začne drugi del sestanka. Le-ta se osredotoča na podrobno načrtovanje uporabniških zgodb. Ekipa izbere uporabniške zgodbe iz seznama uporabniških zgodb, za katere so se zavezali v prvem delu sestanka. Vsaki dodajo potrebne podrobnosti, ki jim bodo pomagale kasneje pri samem razvoju. Če razvijalci dobro premislijo in pripravijo podrobnosti uporabniških zgodb, se v tem primeru uporabniška zgodba med razvojem ne bo drastično spremenila. To pomeni bolj učinkovit razvoj z manj dodatnimi spremembami. Ekipa začne podrobnejše načrtovanje z uporabniško zgodbo na vrhu seznama uporabniških zgodb, ki ima najvišjo prioriteto, in nato nadaljujejo po seznamu navzdol. Po koncu podrobnega načrtovanja mora biti vsakemu članu, ki bo sodeloval pri razvoju, razumljiva vsebina in namen uporabniške zgodbe [48].

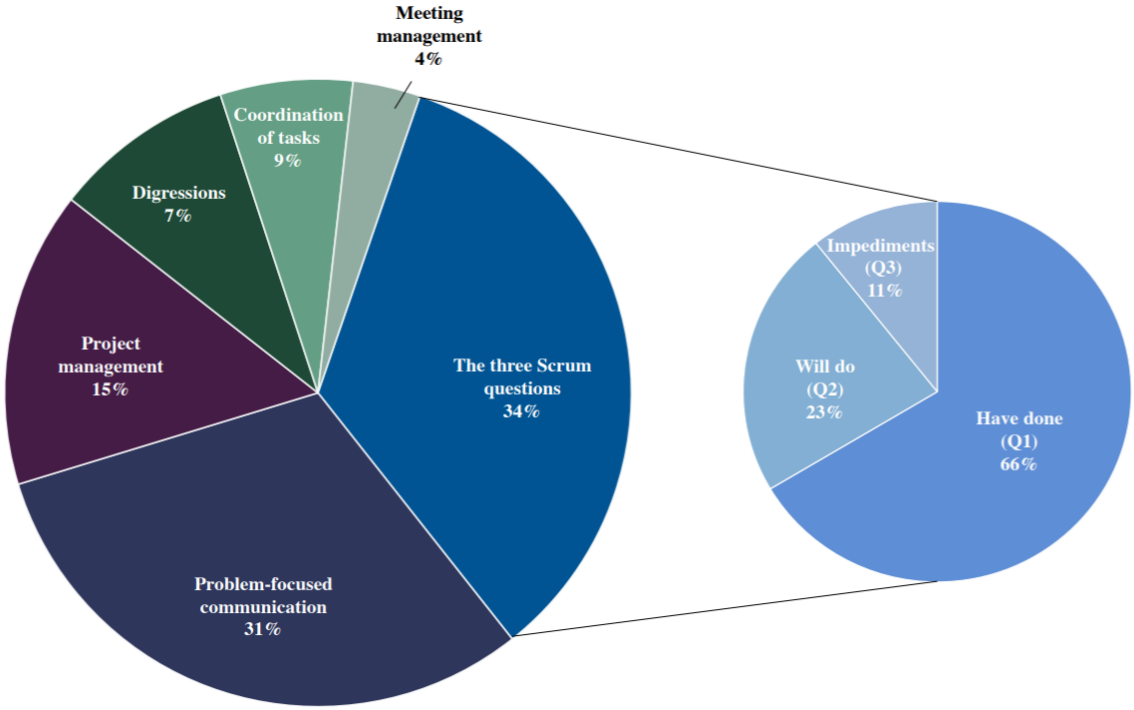
Scrum narekuje, da se morajo člani ekipe sami odločiti, koliko uporabniških zgodb bodo dokončali v naslednji iteraciji, namesto da bi uporabniške zgodbe dodelil lastnik izdelka ali kakšna druga oseba. Na ta način pridemo do zanesljivejše ocene, ki je osnovana na analizi in načrtu razvijalcev in ne s strani tretje osebe. Dejstvo, da morajo razvijalci sami izbirati uporabniške zgodbe za naslednjo iteracijo, je zelo pomembno za motivacijo ekipe, saj se sami zavežejo, da bodo zadani cilj uresničili [48].

2.3.2 Dnevni sestanek

Kot že ime naznanja, se dnevni sestanek zgodi vsak dan (po navadi na začetku delovnega dne). Sestanka se udeležijo lastnik izdelka, skrbnik Scrum projekta in člani razvojne ekipe. Cilj sestanka je usklajevanje dela v ekipi. Člani ekipe med seboj delijo najbolj bistvene podatke, ki so ključni za razvoj projekta (npr. poročajo bistvene novosti o uporabniških zgodbah, na katerih trenutno delajo, s katerimi težavami se soočajo itd.). Člani ekipe drug drugemu nudijo nasvete, kritike in postavljajo vprašanja, ki pomagajo pridobiti nove vpoglede, da se uporabniške zgodbe čim bolj uspešno razvijejo. Prednost dnevnega sestanka je tudi ta, da sodelavci ugotovijo, kako lahko drug drugemu čimbolj pomagajo pri uporabniških zgodbah, katere trenutno razvijajo. Pred sestankom si člani razvojne ekipe postavijo tri vprašanja in na njih poskusijo odgovoriti med sestankom:

1. Kaj sem naredil včeraj, da sem pomagal razvojni ekipi doseči cilj iteracije?
2. Kaj bom naredil danes, da bom pomagal razvojni ekipi doseči cilj iteracije?
3. Ali vidim kakšne ovire, ki mene ali razvojno ekipo ovirajo, da dosežemo cilj iteracije?

Ta sestanek je tudi eden težjih za izpeljavo, saj lahko člani poročajo preveč podrobno. Leto lahko zmanjša učinkovitost sestanka in ga časovno podaljša. Med sestankom je potrebno vzdrževati ravnovesje med pomembnostjo posameznih informacij, ki jih člani ekipe delijo med seboj. Pogosto so razvijalci tisti, ki med sestanki delijo veliko podrobnih informacij, saj se pri delu soočajo z veliko podrobnostmi, katere so za njih zelo pomembne. Načeloma velja pravilo, da se podrobnejše razprave med sestankom prekinejo in se zgodijo po sestanku v manjših skupinah. Na sliki 8 lahko opazimo, kako je čas dnevnega sestanka razporejen v realnem primeru, kar nakazuje, kako težavna je izpeljava dnevnega sestanka. Večina sestanka naj bi bila namenjena trem ključnim vprašanjem, ampak za njih tipična Scrum ekipa po navadi porabi le dobro tretjino dnevnega sestanka.



Slika 8: Časovna razporeditev tem v dnevnem sestanku v primerjavi s porabljenim časom za tri ključna vprašanja, ki bi morala biti bistvo sestanka [45]

2.3.3 Sestanek za izboljšavo seznama uporabniških zgodb (ang. Backlog refinement)

Ena od manj znanih, ampak dragocenih smernic v Scrum-u je, da mora ekipa posvetiti pet do deset odstotkov vsake iteracije za "negovanje" seznama uporabniških zgodb. To vključuje podrobno analizo zahtev, razdelitev večjih uporabniških zgodb na manjše, oceno časa razvoja novih uporabniških zgodb in ponovno oceno obstoječih. Scrum ne narekuje strogo, kdaj in kako je to storjeno, vendar pogosto ekipe uporabljajo sestanek v obliki delavnic blizu konca iteracije. Sestanka se udeležijo lastnik izdelka, skrbnik Scrum projekta in člani razvojne ekipe. Pri dvotedenski iteraciji pet odstotkov pomeni, da za vsako iteracijo obstaja poldnevna delavnica za izboljšavo seznama uporabniških zgodb. V tem sestanku je glavna osredotočenost urediti uporabniške zgodbe za naslednjo ali naslednji dve iteraciji. To omogoča, da je načrtovanje iteracije razmeroma preprosto, saj lastnik izdelka in razvojna ekipa začneta načrtovanje z jasno, dobro analiziranimi in precizno ocenjenimi uporabniškimi zgodbami. Če se ta sestanek ne izvaja ali se izvaja na napačen način, vsebuje sestanek za načrtovanje iteracije pomembna vprašanja, nova odkritja ali zmedo. Le-to vpliva na slabo načrtovanje in potek iteracije [48].

2.3.4 Sestanek za predstavitev rezultatov iteracije (ang. Sprint review)

Ko se iteracija konča, pride čas za tako imenovani sestanek za predstavitev rezultatov iteracije. Tega sestanka se udeležijo lastnik izdelka, skrbnik Scrum projekta, razvojna ekipa, strokovnjaki, vodje ter tudi stranke, za katere razvijamo produkt. Cilj sestanka je pregledati novosti, razvite v iteraciji, in s pogovorom ugotoviti, kje lahko v naslednjih iteracijah izboljšamo pomanjkljivosti. Ključna ideja Scrum-a je pregled in prilagajanje (ang. inspect and adapt). Sestanek za predstavitev rezultatov iteracije pa vključuje metodo pregleda in prilagajanja z osredotočenostjo na produkt. Da to dosežemo, je potrebno prikazati potek dela na projektu, pridobiti mnenja vseh udeležencev in nato razvijati produkt na podlagi povratnih informacij v ponavljajočih se ciklih. Lastnik izdelka tekom sestanka vidi napredek razvoja na projektu, razvojna ekipa pa pridobi nove informacije, kaj se dogaja na trgu. Posledično je najpomembnejši element sestanka za predstavitev rezultatov iteracije poglobljen pogovor med razvojno ekipo in lastnikom izdelka [48].

Pogosto spregledana pomembnost vloge skrbnika Scrum projekta je odgovornost, da vsi poznajo definicijo končanosti. Skrbnik Scrum projekta prepreči, da bi v sestanku za predstavitev rezultatov iteracije prikazali ali razpravljali o uporabniških zgodbah iz seznama uporabniških zgodb, ki ne ustrezajo definiciji končanosti. Uporabniške zgodbe, ki niso dokončane, se vrnejo nazaj v seznam uporabniških zgodb, kjer jih bo lastnik izdelka ponovno razporedil po pomembnosti. Člani razvojne ekipe na ta način ne predstavijo funkcionalnosti, ki ne ustrezajo kakovostnim standardom. S tem se izognejo prikazu programske kode, ki na prvi pogled deluje dobro, ampak je nekakovostno razvita in ni dobro preverjena. Smernica v Scrum-u je, da priprave na ta sestanek ne trajajo več kot 30 minut [48].

2.3.5 Retrospektivni sestanek iteracije (ang. Sprint retrospective)

Retrospektivni sestanek iteracije sledi sestanku za predstavitev rezultatov iteracije in vključuje metodo pregleda in prilagajanja z osredotočenostjo na proces. Sestanka se udeležijo lastnik izdelka, skrbnik Scrum projekta in člani razvojne ekipe. Ta sestanek je namenjen prepoznavanju potencialnih izboljšav in definiranju akcij, ki bodo izboljšale trenutno stanje v Scrum ekipi ali Scrum procesu. Tukaj ima ekipa možnost razpravljati o tem, kaj deluje in kaj ne deluje ter se dogovori, katere spremembe bodo uvedli. Za vsako spremembo je zelo priporočeno definirati eno določeno osebo, ki je odgovorna, da se sprememba uvede na najboljši način [48].

Včasih lahko skrbnik Scrum projekta zelo pripomore pri usmerjanju sestanka v pravo smer, toda še boljše je, če retrospektivni sestanek iteracije vodi drug skrbnik Scrum

projekta, ki dela z drugo ekipo. Na ta način lahko prvotni skrbnik Scrum projekta aktivno sodeluje pri sestanku kot udeleženec in hkrati se znebimo subjektivnosti vodenja sestanka, ki lahko zmanjša možnosti za odkrivanje pomanjkljivosti [48].

Obstaja veliko različnih tehnik vodenja retrospektivnega sestanka iteracije. Veliko število le-teh je opisanih v knjigi Agile Retrospectives [15]. Eden izmed preprostih in pogosto uporabljenih tehnik vodenja retrospektivnega sestanka iteracije je uporaba table z dvema stolpcema, kjer je v prvem zapisano "Kaj je delovalo dobro" in v drugem "Kaj lahko izboljšamo". Vsak udeleženec sestanka lahko doda enega ali več zapisov v kateri koli stolpec. Če se zapisi ponavljajo, jim dodamo kljukice (v primeru pisanja na tablo) ali jih združimo skupaj (v primeru uporabe fizičnih listkov). Na ta način povečamo preglednost elementov na tabli. Priporočeno je, da najprej pregledamo stolpec "Kaj je delovalo dobro", saj s tem vzpodbujamo pozitivno motivacijo med člani ekipe in olajšamo prehod na področje pogovora o tem, kaj lahko izboljšamo. V okvirju pogovora o tem, kaj lahko izboljšamo, poiščemo vzroke, zakaj je prišlo do težav. Oblikujejo se nove delovne naloge (npr. izboljšanje definicije končanosti, optimiziranje sestankov ...), ki se razporedijo med člane ekipe. Pri temu se vsak posameznik odloči za nalogo, katero bo prevzel in poskrbel, da bo opravljena do naslednje iteracije. Na naslednjem retrospektivnem sestanku iteracije člani ekipe ponovno razpravljajo, ali so naloge uspešno opravljene in če so težave še vedno prisotne [48].

3 OPIS, ZAKAJ PODJETJA NE POSPEŠIJO PROJEKTOV, KO ZAČNEJO UPORABLJATI SCRUM

Podjetja se občasno preveč osredotočajo na prednosti Scrum-a in ne dovolj na izzive, ki jih le-ta prinese s seboj. V literaturi sta omenjeni dve fazi, na kateri moramo biti še posebej pozorni:

- vpeljava Scrum-a v podjetje,
- nepravilno spreminjanje osnovnih struktur Scrum-a, ko je Scrum vpeljan v podjetje (t. i. ScrumBut).

V naslednjih poglavjih bomo podrobneje pogledali ta področja.

3.1 IZZIVI PRI VPELJAVI SCRUM-A

Vsaka vpeljava Scrum-a predstavlja veliko novih izzivov za podjetja. Pogosta praksa je, da Scrum vpeljemo postopoma. To pomeni, da od začetka Scrum vpeljemo v le eno ali nekaj manjših ekip v podjetju. Le-te z uporabo praks iz Scrum-a izboljšajo način dela in nato predstavijo prakse iz Scrum ogrodja ostalim članom v podjetju. Na ta način postopoma uvedejo prakse Scrum-a in posledično spodbudijo nove skupine ljudi k prestopu na Scrum. Na primer, Microsoft je pričel z uporabo Scrum-a pri razvoju aplikacije Visual Studio z eno ekipo leta 2008, z več ekipami leta 2009, leta 2010 je bilo 25 ekip, leta 2011 so začeli Scrum uporabljati pri različnih produktih z več sto ekipami in nato so leta 2014 sklenili, da bodo Scrum in ostale agilne metodologije vpeljali v celotnem podjetju. Obstajajo tudi primeri, ko so bile transformacije iz tradicionalnega upravljanja na agilni management izvedene veliko hitreje. Na primer, pri Saleforce.com so leta 2006 izvedli transformacijo v samo treh mesecih. Tukaj moramo omeniti tudi to, da so v podjetju pri projektih že uporabljali in dobro poznali iterativne metode [17].

3.1.1 Izzivi pri vpeljavi Scrum-a v centraliziranem podjetju

Scrum ogrodje ponuja veliko preglednosti, fleksibilnosti in mnogo drugih prednosti. Pri vpeljavi Scrum metodologije v podjetje se soočamo s številnimi izzivi. Eni izmed teh so tudi:

- izzivi v pogledu spreminjanja kulture, strategije in strukture podjetja,
- sprememba vloge projektnega managerja in drugih članov ekipe,
- prilagajanje na nove delovne postopke, orodja in tehnike, ki jih Scrum potrebuje,
- sprememba komunikacije znotraj ekipe in s stranko.

V naslednjih podpoglavjih bomo pogledali vse zgoraj omenjene izzive podrobneje.

3.1.1.1 Izziv v pogledu spreminjanja kulture, strategije in strukture podjetja

Sprememba procesa iz tradicionalnega razvoja v agilnega predstavlja zapletene spremembe v podjetju. Le-te lahko vplivajo na različne vidike v podjetju, ki vključuje njeno strukturo, kulturo in način vodenja. Zato je razumevanje posledic, ki nastanejo zaradi reorganizacije podjetja, prvi korak pri načrtovanju takšnih sprememb.

Ena od največjih sprememb je vpliv Scrum-a na kulturo podjetja, saj ima slednja velik vpliv na:

- procese odločanja,
- strategije reševanja problemov,
- inovativne prakse,
- filtriranje informacij,
- socialna pogajanja,
- odnose,
- mehanizme za načrtovanje in nadzor [38].

Kulturo in miselnost ljudi ni enostavno spremeniti, zaradi česar je premik k agilnim metodologijam za mnoga podjetja toliko večji izziv. Za prestop k agilnim metodam je potreben premik v miselnosti iz upravljajočega načina vodenja (ang. command-and-control management) do načina vodenja na podlagi sodelovanja [5]. Najvišje vodstvo v podjetju se mora naučiti, da agilna preobrazba ni projekt, pobuda, nov postopek ali organizacijska struktura, temveč neprekinjeno potovanje. Le-to je v osnovi drugačno od načina razmišljanja kot pri hierarhičnem pristopu [17]. Organizacijska struktura, ki ima pravo mešanico avtonomije in sodelovanja olajša premik k vpeljavi Scrum-a v podjetje. Na ta način lahko dosežemo prednosti sinergije, hkrati pa zagotovimo prožnost in odzivnost [10].

Pri spreminjanju podjetja iz tradicionalnega v agilno je pomembna robustna strategija, za katero je značilna sposobnost ustvarjanja rezultatov v različnih okoliških razmerah (npr. preučevanje trenutnih prednosti in hitrost prilagoditve, strateška razporeditev trenutne razpoložljivih virov ...). Robustno strategijo sestavljajo trije elementi:

- ekonomska logika,
- močna usmeritev v prihodnost,
- prilagodljivi cilji [56].

<u>Ekonomska logika</u>

Pri tradicionalno organiziranih podjetjih stabilnost vodi v uspešnost s pomočjo učinkovitosti, usklajevanja in rasti. V nasprotju s tem agilna organizacija pričakuje, da bo privedla do uspešnosti s pomočjo preučevanja trenutnih prednosti in s hitrostjo prilagoditve. Ta temeljna ekonomska logika predstavlja pomemben premik v načelih oblikovanja organizacije agilnega podjetja.

<u>Močna usmeritev v prihodnost</u>

Agilna podjetja imajo možnost razviti različne kratkoročne in dolgoročne scenarije za prihodnost. Domneva, da lahko agilno podjetje napoveduje prihodnost, ne obstaja. Ampak takšno podjetje lahko razvije sposobnost preučevanja možnih nepredvidljivih dogodkov in

ga manj presenetijo, kadar se zares zgodijo. Tesen stik z zunanjim okoljem in najnovejšimi trendi zaposlenim v agilnem podjetju omogočajo, da lažje razvijejo različne scenarije, ki se lahko pojavijo v prihodnosti pri izpeljavi projekta. Poleg tega prilagodljivost v agilnem podjetju pripomore pri izvajanju hitrih odločitev kot tudi, da si hitreje opomorejo po slabih odločitvah. Takšen premik v miselnosti zaposlenih iz tradicionalnega podjetja v agilno je težaven in zahteva več časa.

Prilagodljivi cilji

Tradicionalna podjetja izoblikujejo dolgoročne organizacijske cilje. Na začetku načrtovanja ciljev izoblikujejo podroben načrt, kako doseči te cilje, in se ga držijo brez večjih sprememb. Po drugi strani agilna podjetja strateško razporedijo trenutne razpoložljive vire, tako da lahko pridobijo trenutno strateško prednost in so vedno pripravljene na spremembe. Tri dimenzije, ki zagotavljajo fleksibilnost pri prilagajanju ciljev v strategiji agilnega podjetja, so:

- širina, ki se nanaša na paleto ponujenih izdelkov in storitev, število različnih tržišč ali različne tehnologije, ki so temeljne kompetence podjetja;
- agresivnost, ki opisuje nujnost, prizadevnost in sredstva, ki jih podjetje nameni komunikaciji, marketingu in izvedbi svoje strategije, s katero želi pridobiti strateško prednost;
- drugačnost, ki opisuje lastnosti izdelkov in storitev s katerimi se razlikujejo od konkurence (cena, kakovost, garancija, podpora po prodaji in druge značilnosti).

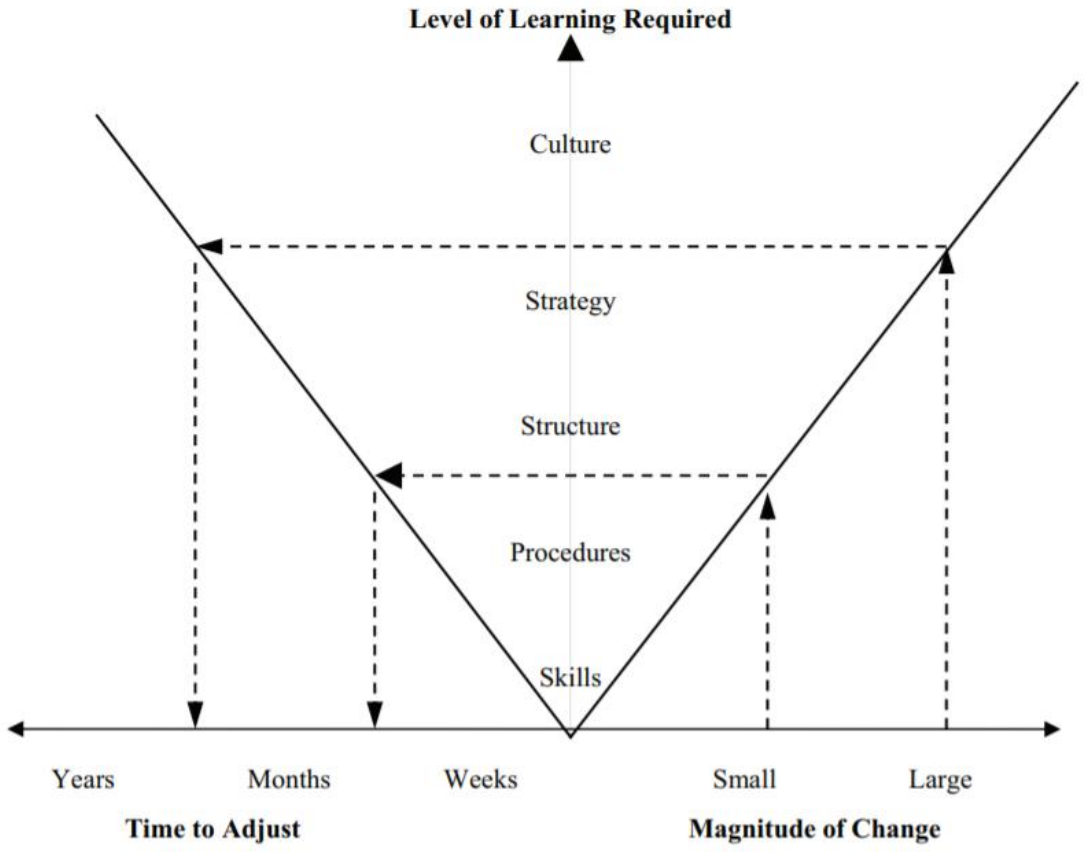
Pri sami strukturi podjetja pa moramo omeniti, da je večina tradicionalnih podjetij veliko bolj optimizirana za učinkovitost kot za strateško okretnost. Hierarhične strukture ter organizacijski procesi, ki smo jih desetletja uporabljali za vodenje in izboljševanje naših podjetij, niso več kos nalogam za zmago v tem hitrejšem svetu [34]. Prestop v agilno podjetje zahteva, da vodstvo premisli o organizacijskih strukturah, funkcijah in uveljavljenih praksah upravljanja, kot so: načrtovanje, oblikovanje proračuna, spodbujevalni in merilni sistemi, ki so globoko usidrani v mentaliteti vodstva [25].

Določena podjetja postopoma vpeljujejo agilne metode ali se ne odločijo, da celotno podjetje spremenijo v agilno. V temu primeru je sama struktura podjetja še bolj kompleksna. Največji izziv je učinkovita izpeljava projektov, kjer določene ekipe delujejo z agilnimi metodami, druge pa še vedno s tradicionalnimi. Boehm in Turner [6] kot največje razlike in izzive pri skupnemu delu agilnih in tradicionalnih metod opazita pri:

- načinu pri dostavljanju funkcionalnosti:
    - agilne: delno dokončane funkcionalnosti,
    - tradicionalne: popolno dokončane funkcionalnosti;
- testiranju funkcionalnosti:

  - agilne: sprotno testiranje funkcionalnosti,
  - tradicionalne: testiranje funkcionalnosti na koncu projekta;
- definiranju zahtev:
  - agilne: neformalne in nepopolne zahteve na začetku projekta,
  - tradicionalne: formalne in popolno definirane zahteve na začetku projekta;
- definiranju arhitekture sistema:
  - agilne: definiranje okvirne arhitekture z vednostjo, kateri deli niso popolnoma definirani,
  - tradicionalne: definiranje celotne arhitekture na začetku projekta.

Ogrodje za organizacijske spremembe, ki sta ga predstavila Adler in Shenhar (1990), je le eno, ki nam pomaga, da si bolje predstavljamo, kako velik je obseg spremembe kulture, strategije in strukture podjetja (slika 9). Tukaj je obseg sprememb velik, raven potrebnega učenja je visoka in čas za prilagoditev na spremembo je dolg. Po drugi strani se na primer tehnološke in procesne spremembe dogajajo na nivoju znanj (ang. skills) in postopkov (ang. procedures). Pri le-teh je za razmeroma majhno spremembo potrebno vložiti malo učenja in čas za prilagoditev je kratek.



Slika 9: Ogrodje za organizacijske spremembe [2]

3.1.1.2 Sprememba vloge projektnega managerja in drugih članov ekipe

Tradicionalna vloga projektnega managerja se po navadi spremeni v vlogo lastnika izdelka ali skrbnika Scrum projekta. Raziskava [39] je pokazala, da je primerneje vlogo projektnega managerja spremeniti v vlogo lastnika izdelka.

Projektni manager se mora spremeniti iz načrtovalca in upravljavca v vlogo moderatorja, ki usmerja in usklajuje skupna prizadevanja tistih, ki sodelujejo pri razvoju. S tem se mora zagotoviti, da se pri končni odločitvi odražajo kreativne ideje vseh udeležencev [28]. Tukaj je največji izziv za projektnega managerja, da se odreče avtoriteti, ki jo je prej užival.

Literatura navaja, da vloga projektnega managerja ne bi smela obstajati v Scrum projektih [49]. Kljub temu je zanimivo dejstvo, da je v večini Scrum podjetij vloga projektnega managerja še vedno prisotna. Raziskava [43] je pokazala, da je v 97 agilnih podjetjih iz 31 držav vloga projektnega managerja še vedno prisotna v 67 % agilnih razvojnih ekip. V raziskavi je bilo razmerje Scrum ekip 58-odstotno. Pri bolj podrobni analizi podatkov so ugotovili, da je možnost prisotnosti projektnega managerja večja v primerih, ko:

- je velikost ekipe med 5 in 10 članov (v primerih, ko ima ekipa 5 ali manj članov, je najmanjša verjetnost, da je vloga projektnega managerja prisotna);
- podjetja implementirajo agilne metode na nivoju ekip, medtem ko nad tem nivojem še vedno potekajo tradicionalne metode projektnega managementa;
- sodelovanje poteka med ekipami na razporejenih lokacijah, kjer je potrebna za usklajevanje porazdeljenih skupin posebna vloga, ki jo izvaja projektni manager.

Shastri, Hoda in Amor [43] pozivajo k dodatnim raziskavam, da se razišče, zakaj je ta vloga še vedno prisotna in kako se je spremenila iz tradicionalne vloge v trenutno vlogo projektnega managerja.

Pri razvojni ekipi lahko vidimo sledeče spremembe, ki so opisane v nadaljevanju. Razvijalci v Scrum podjetju morajo sprejeti večjo odgovornost odločanja, kar pa predstavlja težavo, saj so razvijalci navajeni, da je za odločitve odgovoren projektni manager. Ta miselnost preprečuje razvijalcem, da se zavedajo novih pridobljenih nalog in odgovornosti. Cilj Scrum podjetja je spremeniti miselnost razvijalcev do te mere, da brez pomisleka sprejemajo odločitve, za katere mislijo, da so najboljše za projekt. Ta način dela omogoča izoblikovanje samoorganizirane ekipe, znotraj katere odločitve sprejemajo strokovnjaki, ki imajo največ znanja za določeno področje dela [21].

Spreminjanje miselnosti zaposlenih v podjetju ni enostavno. Zlasti pri izkušenih razvijalcih programske opreme, ki so navajeni dela v tradicionalnem podjetju [12]. Le-ti potrebujejo

dovolj mentorstva in časa, da sprejmejo agilni načina dela (npr. včasih ta dejavnik povzroča težave pri izvajanju programiranja v parih s starejšimi razvijalci) [46]. V fazi tranzicije iz tradicionalnega podjetja v Scrum je še posebej potrebno biti pozoren na ambiciozne razvijalce, ki lahko spremenijo prioritete kupcev izven želja lastnika izdelka. Na ta način otežujejo proces prehoda in sprejetja Scrum metodologije [35].

3.1.1.3 Prilagajanje na nove delovne postopke, orodja in tehnike, ki jih Scrum potrebuje

Ena največjih ovir pri migraciji v agilno podjetje je sprememba procesov v podjetju. Le-ti se morajo spremeniti tako, da podpirajo iterativni razvoj. Uvajanje Scrum metodologije v podjetje vsebuje velike spremembe delovnih postopkov, komunikacijskih kanalov, strategij za reševanje problemov, vlog, orodij in tehnik [38].

Orodja (tako imenovane aplikacije, tehnike, metode ...) igrajo ključno vlogo pri uspešni vpeljavi Scrum-a, še posebej pri razvoju programske opreme. Podjetja morajo vlagati v orodja, ki podpirajo in omogočajo hiter iterativni razvoj, upravljanje različic ter konfiguracij in ostale agilne tehnike. Še bolj pomembno je zaposlene usposobiti za uporabo teh orodij. Potrebno jih je tudi naučiti, kako najbolj učinkovito uporabljati agilne prakse in tehnike, kar naredimo najbolj učinkovito z agilnim usposabljanjem (ang. agile coaching) [38].

Agilno usposabljanje se od drugih metodologij razlikuje, saj je potrebno hkrati upoštevati tako človeški, organizacijski in tehnični pogled.

- Človeški pogled vključuje kognitivne in socialne vidike, ki vključujejo učenje o medosebnih procesih med člani ekipe, stranko in vodstvom.
- Organizacijski pogled vključuje vodstvene in kulturne poglede, ki vključujejo predvsem projektno vodenje in kontroliranje.
- Tehnološki pogled vključuje praktične in tehnične poglede in se nanaša na načrtovanje, testiranje, kodiranje, integracijo, izročitev in vzdrževanje produktov [27].

Agilni trenerji se po navadi osredotočajo na sledeča področja:

- timsko delo in sodelovanje, ki vključuje analiziranje situacij in pogojev v razvojnem okolju, ki vzpodbujajo timsko delo in sodelovanje;
- upravljanje s časom, ki se nanaša na opisovanje agilnih tehnik za časovno upravljanje projektov;

- učenje in refleksija, ki zajema analiziranje razvojnih procesov in pretvorbo le-teh v učne procese z metodo refleksije. Le-ta je uporabljena kot sredstvo, ki podpira te učne procese;
- sprememba in vodstvo, ki vključuje opisovanje načinov za obvladovanje sprememb v projektih in predstavitev kako je vodstvo povezano z upravljanjem sprememb (ang. change management) v agilnih razvojnih okoljih [27].

Agilne ekipe se morajo naučiti in razviti skupno vodstvo ter odločanje. Pri tem imajo pomembno vlogo agilni trenerji, ki imajo veliko izkušenj in znanja z agilnimi praksami. Le-te lahko usposobijo ekipo zaposlenih v učinkovito samoorganizirano agilno ekipo. Hackman, Pallack in Perloff [24] trdijo, da je strokovno usposabljanje za vpeljavo agilnih metodologij in oblikovanje samoorganizacijskih ekip zelo pomembno, saj je brez tega zelo težko v podjetje vpeljati agilno metodologijo. Zaposleni v podjetju so prepoznali koristi in pomembnost usposabljanja za učinkovito sprejemanje agilnih metodologij [1], [26].

Eden izmed največjih izzivov, s katerimi se soočamo pri vpeljavi agilnih metodologij v podjetja, je pomanjkanje izkušenih in potrpežljivih trenerjev (Srinivasan and Lundqvist, 2010). Tudi v raziskavi [46] je le-to prav tako izpostavljeno kot največji izziv.

3.1.1.4 Sprememba komunikacije znotraj ekipe in s stranko

Učinkovita komunikacija je ena najpomembnejših lastnosti za uspešen razvoj [4], [7], [18], [40]. Po mnenju Counsell, Phalp, Mendes in Geddes [14] slaba komunikacija pogosto povzroči hude prekoračitve proračuna in trajanje projekta ter tako vpliva na neustrezno opredeljene zahteve. Zaradi njih se produkt ne razvije kot pričakuje stranka, kar pogosto povzroča pozne spremembe na produktu in vpliva na zamude ter na slabšo kakovost produkta. Vzpostaviti učinkovito komunikacijo nikakor ni lahka naloga. Da bi dosegli učinkovito raven komunikacije, morajo udeleženci biti odprti in z drugimi deliti izkušnje in občutke, ki so jih doživeli pri temi pogovora. Na tak način lahko nadomestijo vrzeli v komunikaciji. Zelo pomembno je tudi upoštevati dejstvo, da teme ne morejo biti vedno razumljene s strani vseh udeležencev [11].

Agilne metode (med njimi tudi Scrum) uporabljajo drugačen pristop pri komunikaciji med vsemi udeleženci. Poudarek je na neformalni, po možnosti osebni komunikaciji [33]. Pri agilnih metodologijah dosežemo neformalno komunikacijo s skupnim pogovorom z vsemi udeleženci. To vključuje tudi stranko, ki je prav tako sestavni del ekipe. Neposredna komunikacija med stranko in ostalimi člani ekipe blaži nesporazume in vzpostavlja obojestransko zaupanje [44].

Pri uporabi agilnih metodologij imajo glavno vlogo pri sprejemanju odločitev stranka in člani ekipe, kar omogoča upoštevanje različnih znanj, stališč, izkušenj in ciljev. To ustvarja tudi večstransko okolje odločanja. Takšen način odločanja je težji kot v primerjavi s tradicionalnim pristopom, kjer je za večino odločitev odgovoren projektni manager. Podjetje mora vložiti ogromno truda, časa in potrpljenja, da lahko ustvari kulturo zaupanja in spoštovanja med zaposlenimi, kar omogoča lažje sodelovanje pri sprejemanju odločitev [10].

Pikkarainen, Haikara, Salo, Abrahamsson in Still [53] so raziskovali izboljšave v komunikaciji pri uporabi različnih agilnih praks (npr. uporaba iteracij, uporaba seznama uporabniških zgodb, sprotno testiranje itd.). V tabeli 1 lahko vidite ugotovitve prednosti agilnih praks pri notranji komunikaciji. V tabeli 2 pa so prikazane prednosti uporabe agilnih praks pri zunanji komunikaciji.

Tabela 1: Notranja komunikacija

Agilna praksa	**Prednosti**
Odprti pisarniški prostor	Potreba po dokumentaciji se lahko zmanjša, saj vsi v projektu poznajo skupne cilje in njihov status.
Dnevni sestanki	Dnevni sestanki so za Scrum ekipo (v določenih primerih tudi za stranko) dobra praksa za seznanjanje o statusu napredka dela na uporabniških zgodbah v projektu.
Tabla z nalogami	Podatki o statusu projekta so na voljo na tabli, kar omogoča boljši pregled projekta.
Iterativno (ponavljajoče v vsakem Scrum ciklu) načrtovanje iteracije	Celotna ekipa se zaveda načrtov projekta in cilja naslednje iteracije.
Iterativni retrospektivni sestanki iteracije	Učinkovita praksa za izboljšanje in uporabo agilnih praks.
Programiranje v paru	Učinkovita praksa za pregled kode (predlagana je neobvezna uporaba, saj je na dnevni osnovi ta praksa težko učinkovito izvedljiva).
Kontinuirana integracija (ang. continuous integration)	Učinkovita praksa za sporočanje trenutnega stanja projekta ljudem, ki izvajajo testiranje. Ta praksa olajša testiranje.

Tabela 2: Zunanja komunikacija

Agilna praksa	**Prednosti**
Iterativno načrtovanje iteracij	Sistematičen način izmenjave informacij med vsemi udeleženci. Izboljšana preglednost nad kratkoročnimi cilji projekta. Uporaba jasno opredeljenih uporabniških zgodb pomaga stranki, da se približa razvoju in na ta način lažje pove, katere zahteve so za njih najbolj pomembne.
Iterativni sestanek za predstavitev rezultatov iteracije	Sistematični način prikazovanja končanih funkcionalnosti in izmenjave informacij med vsemi udeleženci. Izboljšana preglednost nad kratkoročnimi cilji.

Korkala, Abrahamsson in Kyllönen [31] so ugotovili, da kadar stranka ni prisotna na sestankih, kjer se definirajo naslednji koraki za razvoj produkta, se poveča število kasnejših sprememb v projektu. Kasnejšim spremembam se je mogoče izogniti s pogostejšo komunikacijo s stranko in več povratnimi informacijami. Ugotovili so tudi, da je potrebno več pozornosti nameniti izbiri ustreznih komunikacijskih medijev (npr. različni video in telefonski konferenčni sistemi), ki so uporabljeni med iteracijami, kadar se stranka ne more fizično udeležiti sestanka.

Korkala, Pikkarainen in Conboy [32] so primerjali sodelovanje agilnega in tradicionalnega oddelka v istem podjetju na različnih krajih. V raziskavi je agilni oddelek razvijal produkt, medtem ko je tradicionalni oddelek podajal zahteve za razvoj produkta. Tradicionalni oddelek se je v tem primeru vedel kot stranka v odnosu do agilnega oddelka. Ugotovili so, da kadar ni prisotno zaupanje med oddelki in ko stranka (tradicionalni oddelek) ne sodeluje aktivno, je razvoj zelo zapleten.

3.1.2 Izzivi pri vpeljavi Scrum-a v porazdeljenem podjetju

Pri vpeljavi Scrum-a v porazdeljenem podjetju so izzivi, ki smo jih omenili v poglavju 3.1.1, še večji. Sama prioriteta izzivov je tudi drugačna, saj največji izziv predstavljajo komunikacijske in kulturne razlike. V takšnih podjetjih je zaradi oddaljenosti težko izvesti sestanek iz oči v oči, ki je v agilnem manifestu omenjena kot najbolj učinkovita komunikacijska metoda. Dodatna ovira pri komunikaciji je tudi časovna razlika krajev [20].

Kulturna razlika je že velik problem v zahodnih državah, še toliko večja pa je, ko na primer sodelujemo s podružnico podjetja iz drugega dela sveta. V nadaljevanju je opisan primer sodelovanja dveh oddelkov iz evropske in azijske države. Na primer v azijskih kulturah vodstvo odvrača ljudi od postavljanja vprašanj, pogovarjanja o problemih, opozarjanja o neizvedljivih časovnih rokih ali predlaganju alternativ. Evropske ekipe se morajo tega

dobro zavedati in upoštevati pri svojem delu. Evropska ekipa se mora umakniti nazaj, ko zaznajo, da se azijska ekipa le pasivno strinja z odločitvami. Slaba novica pri tem je, da je pridobivanje proaktivnosti pri takšnih ekipah težaven boj in neizogibno traja dlje časa. Praksa narekuje, da se ne sme domnevati, da bo azijska ekipa omenila težave, tudi ko so opažene. Prilagajanje na Scrum način dela v azijskih ekipah je težji in traja dlje časa, saj ljudje niso navajeni na sodelujoč način dela. Po drugi strani, ko zaposleni v azijskih podjetjih spoznajo, da imajo svobodo in odgovornost pri sprejemanju odločitev, pri tem zelo uživajo. Avtonomija je odličen motivator, ki omogoča, da so ljudje bolj produktivni. Hkrati člani ekipe bolj samozavestno in z večjim zaupanjem sprejemajo odločitve in ne čakajo na odobritev članov druge ekipe, kar vodi do manjših časovnih zamud pri projektih [20].

3.1.3 ScrumBut oz. nepravilno prilagajanje Scrum-a

Za uspešno uporabo Scrum-a je potrebno imeti temu prilagojeno strukturo podjetja in pravilno postavljene Scrum procese, ki omogočajo tekoče delo na projektih. Izvajanje Scrum-a v podjetjih le redko deluje, kot je zapisano v literaturi. Podjetja pogosto odstopajo od predlaganih praks iz Scrum-a zaradi različnih razlogov. Četudi so nekatera odstopanja razumna, določena podjetja prilagajajo prakse iz Scrum-a brez jasnega razumevanja, do kakšnih negativnih posledic lahko to privede. To lahko zmanjša uspešnost projektov po uvedbi Scrum-a [19].

Na Scrum moramo gledati kot na celovito ogrodje in ne le kot skupek praks, saj so prilagajanja potencialno škodljiva. To seveda ne pomeni, da Scrum-a ne smemo prilagoditi našim potrebam v podjetju, saj lahko določene spremembe dvignejo uspešnost dela in so zelo koristne. Pomembno je biti dobro seznanjen, kaj se lahko zgodi, če spremenimo določen del Scrum ogrodja. Med drugim je potrebno tudi hitro odreagirati, če sprememba ne vodi v pravilno oz. načrtovano smer. Da bi se izognili morebitnim težavam, je potrebno dobro poznati posledice prilagajanja Scrum-a in to upoštevati pri vpeljavi ali prilagajanju ogrodja [19].

Samo ogrodje Scrum-a je enostavno za razumeti, vendar je izvajanje projektov po Scrum pravilih težko izvedljivo v praksi. Ljudje se pogosto skušajo izogniti strogo predpisanim praksam iz Scrum-a in prilagodijo okvir Scrum-a po njihovih željah brez razumevanja posledic. To je izhodišče ScrumBut sindroma [36].

Opredelitev ScrumBut-a:
ScrumBut ima posebno obliko: (ScrumBut) [razlog] {rešitev}

Primer: „(Uporabljamo Scrum, vendar) [vsakodnevni Scrum sestanki vzamejo preveč časa,] {zato jih imamo samo enkrat na teden, razen če jih potrebujemo pogosteje}“ [36].

V temu poglavju se bomo posvetili pregledu in razumevanju najpogostejših napak pri prilagajanju Scrum-a. Podrobneje bomo pogledali, kaj so prvotni razlogi za odstopanje od Scrum praks in tudi kakšni so vplivi na delo projektnih ekip.

V obširni raziskavi [16], ki je upoštevala rezultate 17 raziskav in 9 intervjujev, so ugotovili glavne razloge, zakaj se zgodi ScrumBut. Razloge so razdelili na sledeče kategorije:

- vloge v Scrum-u,
- Scrum dogodki,
- preostali razlogi.

V tabelah 3 in 4 bomo predstavili glavne razloge za ScrumBut v povezavi z vlogami v Scrum-u. Glavne razloge v povezavi z vlogo lastnika izdelka in skrbnika Scrum projekta lahko vidite v tabeli 3, glavne razloge v povezavi z razvojno ekipo lahko vidite v tabeli 4.

Tabela 3: Razlogi za ScrumBut v povezavi z vlogo lastnika izdelka in skrbnika Scrum projekta

Priporočena Scrum praksa	**ScrumBut**	**Razlog za ScrumBut**
Lastnik izdelka je ena oseba in ne več oseb.	SB1: Imeti več lastnikov izdelka.	Ljudje z različnimi deleži želijo avtoriteto.
		Odločevalci se ne želijo odreči mandatu.
		Razvoj izdelka poteka na več podružnicah.
		Več različnih odnosov s strankami.
Lastnik izdelka je odgovoren za urejanje uporabniških zgodb v seznamu uporabniških zgodb.	SB2: Lastnik izdelka ni odgovoren za urejanje uporabniških zgodb v seznamu uporabniških zgodb.	Upravni odbor želi biti odgovoren za smer razvoja produkta.
Scrum ekipo sestavljajo lastnik izdelka, skrbnik Scrum projekta in razvojna ekipa.	SB3: V ekipi ni lastnika izdelka.	Naloge lastnika izdelka opravljajo prejšnje vloge iz tradicionalne organizacije.
		Naloge lastnika izdelka lahko reši ekipa sama (brez lastnika izdelka).
		Vloga lastnika izdelka je odvečna zaradi vlog projektnega vodje in produktnega managerja.

		Zahteve so prejete neposredno od stranke.
	SB4: V ekipi ni skrbnika Scrum projekta.	Lažje je sodelovati z ekipo brez skrbnika Scrum projekta.
		Vlogo skrbnika Scrum projekta je prevzel projektni manager.
		Bolj smiselno je uporabiti proračun za razvojno ekipo.
		Ekipa ne uporablja zadosti Scrum dogodkov, da bi upravičila vlogo skrbnika Scrum projekta.

Skrbnik Scrum projekta in lastnik izdelka sta ključna člana Scrum ekipe. Kljub temu se ekipe odločajo, da teh vlog ne uporabljajo v Scrum procesu, kot je razvidno iz tabele zgoraj. Najpogosteje vlogo lastnika izdelka prevzamejo druge vloge ali ekipa pridobiva zahteve neposredno od stranke. Pogosto pa je prisotnih več lastnikov izdelka, saj želi več oseb imeti končno odločitev. Vsak od njih misli, da poznajo najboljši način, kako razviti produkt in kako s tem najbolj pomagati podjetju. Ekipe uporabljajo več lastnikov izdelka tudi v primeru, ko eden ne uspe opraviti vseh svojih nalog in se zato razvije skupina lastnikov izdelka. Po drugi strani vlogo skrbnika Scrum projekta večinoma ekipe ne uporabljajo, saj to vlogo prevzamejo druge vloge ali vloga skrbnika Scrum projekta ni ocenjena kot zadosti koristna [16].

V raziskavi [19] so odkrili, da določena podjetja uporabljajo enega lastnika izdelka v različnih ekipah, kjer je bila največja težava, da lastnik izdelka ni imel zadosti časa za kreiranje zahtev (SB5). V določenih primerih so lastniki izdelka predstavniki stranke, ki pa delujejo izven okvirja ekipe in ne razumejo dinamike razvoja produkta. Odgovornost lastnika izdelka je med drugim povečanje donosnosti naložb (ang. Return On Investment), kar v temu primeru ni mogoče, saj lastnik izdelka dela za stranko (SB6).

V tabeli 4 bomo predstavili razloge za ScrumBut v povezavi z razvojno ekipo.

Tabela 4: Razlogi za ScrumBut v povezavi z razvojno ekipo

Priporočena Scrum praksa	**ScrumBut**	**Razlog za ScrumBut**
Velikost razvojne ekipe mora biti med pet in devet članov.	SB7: Preseganje mej velikosti ekipe.	Ekipa dela na projektu, ki ne potrebuje veliko sredstev.
		Podjetje je zelo novo.

		Ekipa je bila velika, še preden je bil Scrum vpeljan.
Člani razvojne ekipe v Scrum-u so razvijalci, drugih nazivov ne poznamo.	SB8: Drugi nazivi so uporabljeni pri razvojni ekipi.	Olajšano upravljanje zaposlenih.
		Arhitekti so visoko usposobljeni in imajo veliko izkušenj.
		Želja vodstva v podjetju je imeti druge nazive, saj je to del identitete ljudi v podjetju.
		Ljudem olajša prehod na različne oddelke.
Razvojne ekipe so večfunkcijske (ang. cross-functional).	SB9: Razvojna ekipa ni večfunkcijska.	Razvoj je večkrat odvisen od starejše komponente.
		Obstaja ločena skupina za testiranje.

V tabeli 5 bomo predstavili razloge za ScrumBut v povezavi s Scrum dogodki.

Tabela 5: Razlogi za ScrumBut v povezavi s Scrum dogodki

Priporočena Scrum praksa	**ScrumBut**	**Razlog za ScrumBut**
Dnevni Scrum sestanki se izvajajo vsakodnevno.	SB10: Dnevni sestanki so preklicani ali preloženi.	Dnevni sestanki potekajo samo takrat, ko skrbnik Scrum projekta potrebuje posodobitve.
		Pogosta srečanja z zunanjimi sodelavci onemogočajo izvedbo sestanka.
		Zaradi tesnega sodelovanja so dnevni sestanki odvečni.
		Člani ekipe se težko srečujejo vsak dan.
		Včasih član ekipe ne pride pravočasno na sestanek.
		Napredek je premajhen, da bi bil sestanek smiseln.
		Ekipa je zelo majhna.
Retrospektivni sestanek iteracije se izvaja na koncu vsake iteracije.	SB11: Retrospektivni sestanki iteracije so preklicani, preloženi ali niso načrtovani.	Ljudje, predvsem lastnik izdelka, se ne morejo udeležiti sestanka.
		Težave rešujemo takoj, ko se pojavijo.

		Imamo premalo tem za pogovor v vsaki iteraciji.
		Časovni pritisk, ki ga narekuje nekdo izven ekipe (npr. vodstvo).
		Ekipa je prevelika.
		Pomanjkanje tem za pogovor v sestanku.
		Probleme rešujemo na drugih sestankih.
Sestanki za predstavitev rezultatov iteracije se izvajajo na koncu vsake iteracije.	SB12: Sestanki za predstavitev rezultatov iteracije so preklicani ali preloženi.	Dober odnos s strankami in pogosti stiki naredijo sestanke za predstavitev rezultatov iteracije nepotrebne.
		Tedensko srečanje naredi sestanke za predstavitev rezultatov iteracije nepotrebne.
Sestanki za načrtovanje iteracije se izvajajo na začetku vsake iteracije.	SB13: Sestanki za načrtovanje iteracije so preklicani ali preloženi.	Ne uporabljamo ocen, zato poteka načrtovanje na zahtevo.
		Iteracije se nadomestijo s Flow metodo iz Lean metodologije.
		Ljudje, predvsem lastnik izdelka, se ne morejo udeležiti sestanka.
Srce Scrum-a je iteracija.	SB14: Zamenjava iteracije s Flow metodo iz Lean metodologije.	Trajanje projekta je zelo kratko.
		Potrebno je biti zelo odziven na zahteve strank.
	SB15: Ekipa ne uporablja iteracij.	Trajanje projekta je zelo kratko.
		Potrebno je biti zelo odziven na zahteve strank.
Retrospektivni sestanek iteracije se izvaja po sestanku za predstavitev rezultatov iteracije.	SB16: Združevanje obeh sestankov.	Ločevanje sestankov se zdi nesmiselno.
		Ločena srečanja trajajo preveč časa.
		Retrospektivni sestanek iteracije ni načrtovan, zato je dnevni red sestanka dodan sestanku za predstavitev rezultatov iteracije.

Celotna razvojna ekipa mora sodelovati pri sestankih.	SB17: Več dnevnih sestankov z različnimi udeleženci.	Razvojna ekipa je prevelika, da bi vključili vse v enemu sestanku.
Udeleženci sestanka za predstavitev rezultatov iteracije so člani ekipe in ključne interesne skupine (npr. stranka).	SB18: Sestanek za predstavitev rezultatov iteracije poteka brez ključnih interesnih skupin.	Ključne interesne skupine nimajo časa za sodelovanje.
		Sestanka se udeležijo druge ekipe v podjetju.
		Lastnik izdelka deluje v imenu stranke.
		Sestanki so predolgi in preveč podrobni.
Dnevni sestanki so dogodki, namenjeni razvojni ekipi.	SB19: Nekateri razvijalci se ne udeležujejo dnevnih sestankov.	Kombinacija zamud udeležencev in ne osredotočenih razprav v dnevnem sestanku.
Dnevni sestanek je dogodek, ki traja največ 15 minut.	SB20: Podaljševanje trajanja dnevnega sestanka.	Podrobne razprave so dovoljene med dnevnim sestankom.
		Dnevni sestanek ne poteka vsak dan.
Ko se iteracija začne, je trajanje določeno in ga ni mogoče skrajšati ali podaljšati.	SB21: Podaljševanje trajanja iteracije med iteracijo.	Uporaba novih tehnologij brez predhodnega usposabljanja.
Sestanek za načrtovanje iteracije je časovno omejen na največ osem ur za enomesečno iteracijo.	SB22: Podaljšanje trajanja sestanka za načrtovanje iteracije.	Ocenjevanje težavnih uporabniških zgodb traja dlje časa zaradi nezmožnosti dogovora o oceni kompleksnosti.

Eden najbolj pogostih ScrumBut-ov je prestavljanje ali odpovedovanje Scrum dogodkov. Pri vseh Scrum dogodkih lahko vidimo problem udeležbe članov oziroma zamujanja članov. Ekipe imajo največ težav z dnevnimi sestanki, kjer je glavni namen doseči prenos bistvenih informacij med člani ekipe, da lažje dosežemo cilj iteracije. V dnevnih sestankih člani ekipe prepogosto izmenjujejo nepomembne informacije za druge udeležence. Večinoma gre za posodobitve statusa dela na uporabniških zgodbah, ki so prepodrobne za druge člane ekipe ali za informacije, ki so bistvene le za vodstvo. V sestanku tudi prepogosto dovolimo podrobne razprave, ki bi se morale zgoditi takoj po koncu sestanka v manjših skupinah in se zaradi tega sestanek pogosto podaljša. Glavni namen dnevnega sestanka je pogosto pozabljen in zaradi tega udeleženci ne vidijo prednosti sestanka ali mislijo, da je sestanek prepogost. Tesno sodelujoče (predvsem manjše) ekipe menijo, da je sestanek nepotreben, saj se že vse vnaprej dogovorijo [16].

Pogosto člani ekipe menijo, da retrospektivni sestanek iteracije ni potreben, saj že spotoma rešujejo težave, ko se pojavijo. Določene ekipe omenjajo tudi, da imajo premalo tem za

pogovor, da bi bil sestanek smiseln. Retrospektivni sestanek iteracije je pogosto vključen v dnevni red drugih sestankov, ko ekipe menijo, da njihov Scrum model deluje zelo učinkovito in ni potrebe po ločenem sestanku za obširnejši pogovor o tej tematiki [16].

ScrumBut ni pogosto opažen pri sestanku za predstavitev rezultatov iteracije. Le-ta se zgodi v primerih, ko ekipa misli, da je stranka že tako zelo vključena pri razvoju ali ko stranka ne dojema, da je sestanek vreden njihovega časa [16].

V enem primeru se je ekipa odločila, da lahko trajanje iteracije podaljšajo med trajanjem iteracije. Do tega zaključka so prišli, saj so začeli uporabljati nove tehnologije in niso vedeli, katera dolžina iteracije je najbolj primerna za njih. Na ta način so želeli pridobiti osredotočenost s spopadanjem s težavami, ki so povezane z novo tehnologijo, in niso želeli, da bi se ekipa soočala z nepotrebnimi frustracijami, ki bi izhajale iz velikega števila nezaključenih iteracij [16].

V tabeli 6 bomo podrobneje pogledali preostale razloge za ScrumBut.

Tabela 6: Preostali razlogi za ScrumBut

Priporočena Scrum praksa	**ScrumBut**	**Razlog za ScrumBut**
Samo razvojna ekipa lahko spremeni vsebino seznama nalog iteracije med potekom iteracije.	SB23: Vodstvo spreminja seznam nalog iteracije med potekom iteracije.	Vodstvo zahteva višjo prioriteto za določene uporabniške zgodbe in popravke napak.
Razvojna ekipa je odgovorna za vse ocene velikosti uporabniških zgodb.	SB24: Razvojna ekipa ne ocenjuje velikosti uporabniških zgodb.	Dovolj je, da pri oceni vključimo tiste z domenskim znanjem.
		Ocena stroškov produkta je potrebna vnaprej.
		Ocenjevanje traja predolgo.
Uporabniške zgodbe za seznam nalog iteracije izbere razvojna ekipa.	SB25: Uporabniške zgodbe za seznam nalog iteracije izbere vodstvo.	Navada iz starega hierarhičnega razvojnega procesa.
Seznam uporabniških zgodb mora biti viden, pregleden in jasen vsem.	SB26: Seznam uporabniških zgodb ni pregleden.	Več lastnikov izdelka vzdržuje svoj del seznama uporabniških zgodb.
Seznam uporabniških zgodb je urejen seznam vsega, kar je potrebno vključiti v	SB27: Neustrezni elementi seznama uporabniških zgodb niso odstranjeni.	Odgovorni ljudje ne vedo, ali bo določen element potreben ali ne.

produkt.		Neprestano posodabljanje seznama uporabniških zgodb vzame preveč časa in le najpomembnejše uporabniške zgodbe na vrhu so urejene.
Seznam uporabniških zgodb je urejen seznam.	SB28: Seznam uporabniških zgodb ni urejen.	Za ureditev seznama uporabniških zgodb je potrebno vložiti preveč časa in truda.
		Ni lastnika izdelka.
		Lastnik izdelka nima zahtevanega znanja.
Seznam uporabniških zgodb je ustvarjen in je v uporabi.	SB29: Seznam uporabniških zgodb ne obstaja.	Seznam uporabniških zgodb ni bil nikoli ustvarjen.
Elementi seznama uporabniških zgodb, ki so izbrani v seznam nalog iteracije, morajo biti opravljeni znotraj iteracije.	SB30: Razvoj uporabniških zgodb traja dlje od trajanja iteracije	Študenti delajo le za krajši delovni čas.
		Nekatere uporabniške zgodbe zahtevajo veliko raziskav/eksperimentiranja.
Vsak sistem ali produkt ima definicijo končanosti, ki je standard za kakovost opravljenega dela v uporabniških zgodbah.	SB31: Definicija končanosti ne obstaja.	Kriteriji o končanosti so določeni pri vsaki uporabniški zgodbi posebej.
Razvojna ekipa dostavlja nove funkcionalnosti vsako iteracijo.	SB32: Na koncu iteracije ni povečanja funkcionalnosti na produktu.	Iteracije se izmenično spreminjajo med iteracijami, ki dostavljajo nove funkcionalnosti in iteracijami, ki odpravljajo napake. Tako se nove funkcionalnosti dodajo vsako drugo iteracijo.
		Nove funkcionalnosti se dostavijo takoj, ko so narejene.

Velikokrat lahko opazimo ScrumBut zaradi vpletenosti vodstva v Scrum procese. Vodstvo se vpleta pri zahtevah in razvrščanju uporabniških zgodb, kar vpliva na pogoste spremembe začetnega načrta iteracije in otežuje ekipi, da doseže zastavljeni cilj. Številne ScrumBute lahko najdemo v povezavi s seznamom uporabniških zgodb, saj je le-ta velikokrat nepregleden in neurejen. Glavni razlogi za to so: urejanje vzame preveč časa, ureja ga več lastnikov izdelka ali lastnik izdelka nima pravilnih znanj, da ga lahko uredi in

potrebuje pomoč razvojne ekipe. V primeru, da lastnik izdelka ne obstaja v ekipi, je urejanje seznama uporabniških zgodb še toliko težje, saj drugi člani ekipe nimajo znanja, da določijo, ali je določen element potreben v seznamu ali ne. V določenih primerih seznam uporabniških zgodb ne obstaja, saj ga ekipa ni definirala na začetku projekta. Določene ekipe v raziskavi, ki sta jo izvedla Dræger in Petersen [16], menijo, da se celotni razvojni ekipi ni potrebno udeležiti ocenjevanja velikosti elementov iz seznama uporabniških zgodb. Menijo, da v tem primeru zadostuje prisotnost oseb z domenskim znanjem, saj drugače traja ocenjevanje predolgo. Vidimo, da razvoj uporabniških zgodb pogosto traja dlje od trajanja iteracije, ko le-te vsebujejo veliko raziskovanja in preizkušanj ali ko na njih dela zaposleni, ki ni zaposlen za polni delovni čas. V obeh primerih je trajanje uporabniške zgodbe težko pravilno oceniti. Nove funkcionalnosti pogosto niso dostavljene vsako iteracijo, ker so dostavljene takoj, ko so narejene ali ker se iteracije za nove funkcionalnosti in test izmenično spreminjajo [16].

V raziskavi [19] so ugotovili sledeča ScrumButa v povezavi z ocenjevanjem velikosti uporabniških zgodb: ekipa ne ocenjuje velikosti uporabniških zgodb (11 odstotkov ekip) in oceno dodeli lastnik izdelka ali projektni manager (33 odstotkov ekip). Slednje nakazuje, da lastnik izdelka ali projektni manager narekuje, katere uporabniške zgodbe bodo implementirane v naslednji iteraciji, kar je naloga razvojne ekipe. Določene ekipe ne uporabljajo padajočega diagrama spremljanja napredka (39 odstotkov ekip), saj ne pridobijo koristnih informacij iz diagrama (SB33). V primeru, ko je uporabljen padajoči diagram spremljanja napredka, se tudi zgodi, da ga lastnik izdelka ali projektni manager riše zase in ni viden celotni ekipi (SB34). Pri evaluaciji samoorganiziranosti ekip so bili rezultati sledeči: 50 odstotkov ekip je samoorganiziranih in razvojna ekipa se odloča, na katerih uporabniških zgodbah delajo, v preostalih 50 odstotkih lastnik izdelka ali projektni manager dodeljuje uporabniške zgodbe razvojni ekipi (SB35) [19].

Pavlič in Brezočnik (2017) sta v njihovi raziskavi [51] opazila še spodaj opisane ScrumBut-e, ki se pogosto zgodijo pri uporabi Scrum-a:

- Pretirana uporaba metrik (npr. spremljanje ključnih kazalcev učinkovitosti – KPI) povzroči, da se ekipa osredotoča na delo, ki bo prikazalo dobre kazalce in ne na delo, ki bo prineslo največ vrednosti za stranko. Pomembno je, da podjetja izvajajo meritve kazalnikov učinkovitosti neinvazivno, razvojnim ekipam pa naj bo jasno razložen namen merjenja, saj drugače lahko ekipa meni, da je namen merjenja kaznovanje, če uspešnost ni dosežena (SB36).
- Uporabniške zgodbe so opredeljene horizontalno in ne vertikalno, kjer horizontalno pomeni, da se razvije samo določen sloj implementacije (npr. delo samo na podatkovni bazi). V naslednjih iteracijah se še razvijejo posamezno tudi drugi sloji (npr. implementacija na vmesnem nivoju, implementacija uporabniškega vmesnika

...). To je v nasprotju z agilnimi principi, kjer vsaka implementacija pomeni dodano vrednost za končnega uporabnika. Vertikalni razvoj pomeni, da se vsak sloj razvije delno, ampak funkcionalnost, ki povezuje vse sloje, prinese dodano vrednost za končnega uporabnika (SB37).

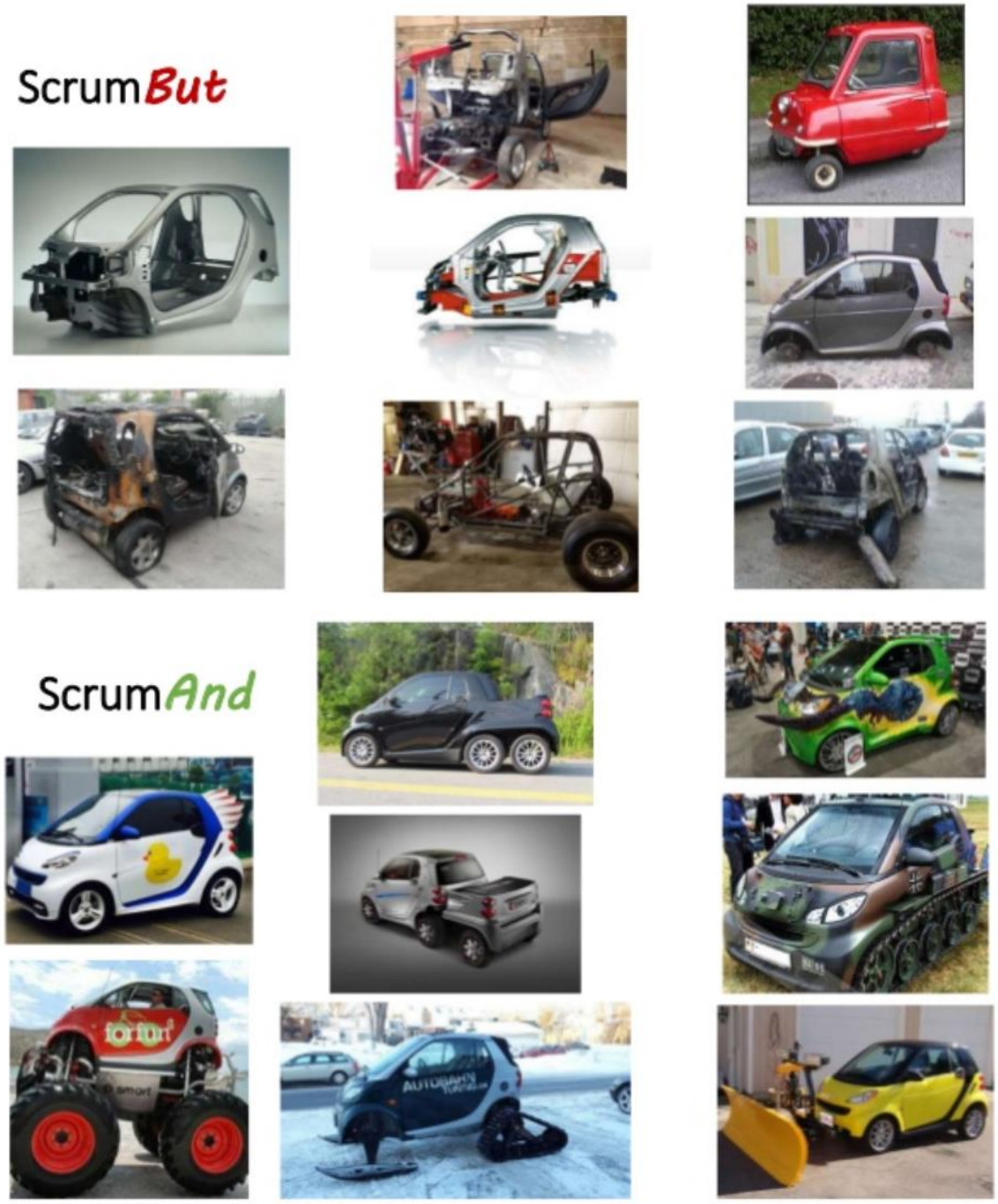
3.2 SCRUMAND OZ. DODAJANJE PRAKS DRUGIH METODOLOGIJ K OSNOVNI STRUKTURI SCRUM-A

V prejšnjem poglavju smo omenili težave s prilagajanjem Scrum praks (ScrumBut). Četudi se je potrebno izogibati neprimernemu prilagajanju Scrum praks, lahko razširitev Scrum praks prinese boljše rezultate. Na primer k standardnim Scrum praksam lahko dodamo prakse ekstremnega programiranja (ang. extreme programming) ali kakšne druge metodologije. Takšno razširitev Scrum praks imenujemo ScrumAnd [16].

Čeprav je princip ScrumAnd-a splošen in je lahko uporabljen v kombinaciji z različnimi metodologijami, je najpogosteje omenjena integracija Scrum-a s praksami ekstremnega programiranja. Takšna integracija lahko podjetju prinese inovativne pristope pri delu porazdeljenih ekip in spodbujanju hitrega razvoja, kar lahko sčasoma ustvari zelo produktivno Scrum okolje. Najpogosteje uporabljene prakse iz ekstremnega programiranja v kombinaciji s Scrum-om so: testno voden razvoj (ang. test driven development), parno programiranje (ang. pair programming) in stalno povezovanje (ang. continuous integration). Razširitev Scrum praks je tudi uspešno izvedena s praksami, kot so CMMi, PMP, Prince, ISO itd. Takšne razširitve pogosto vidimo v primeru, ko morajo podjetja zadostiti pogojem določenega industrijskega standarda [36].

Scrum nam ponuja svobodo gibanja, da razširimo ogrodje z drugimi praksami. Pri tem morajo podjetja previdno izbrati in uporabiti najbolj primerne prakse iz poslovnega sveta glede na njihove potrebe in cilj (npr. upravljanje tveganj, nadzor sprememb ...) [36].

Razlike med ScrumBut-om in ScrumAnd-om v prenesenem pomenu lahko vidite na sliki 10.



Slika 10: Razlika med ScrumBut-om in ScrumAnd-om [23]

4 POSLEDICE IN PREDLOGI ZA IZBOLJŠAVO SCRUMBUTOV

V tem poglavju se bomo osredotočili, da za vsak prej omenjeni ScrumBut opišemo, kakšne posledice lahko prinese ter kakšni so predlogi za izboljšavo.

Pri tem bi najprej želeli omeniti, da tukaj niso omenjene posledice in predlogi za rešitev ScrumBut-a SB14: »zamenjava iteracije s Flow metodo iz Lean metodologije«. Odstopanje od osnov Scrum-a je preveliko in zelo vpliva na celoten proces Scrum-a. Podatki iz literature nam ne podajo jasnih posledic in dejstev, na kaj moramo biti pozorni, oziroma predlogov, kako bi situacijo izboljšali.

4.1 SB1: IMETI VEČ LASTNIKOV IZDELKA

Tabela 7: Posledice in predlogi za izboljšavo SB1

Možne posledice	**Predlogi za izboljšavo**
Cilj iteracije ni jasen razvojni ekipi, kar lahko vodi do neenotnega oz. manj učinkovitega dela in nedoseganja cilja iteracije. Razvojna ekipa nima jasnega pogleda na celoten produkt in izgubi osredotočenost, kaj prinese največjo vrednost za stranko. To lahko povzroči, da se najpomembnejše funkcionalnosti ne razvijejo najprej. Omenjene posledice so ozko povezane s ScrumBut-i: • seznam uporabniških zgodb ni pregleden (SB26), • neustrezni elementi seznama uporabniških zgodb niso odstranjeni (SB27) in • seznam uporabniških zgodb ni urejen (SB28). Prednostne naloge so določene za posamezen del razvojne ekipe in ne za celoten produkt, kar lahko privede, da se ne razvijejo najpomembnejše funkcionalnosti najprej. Seznam uporabniških zgodb ni urejen ali pregleden, kar lahko povzroči podaljševanje sestanka za načrtovanje iteracije ter sestanka za izboljšavo seznama uporabniških zgodb. Omenjene posledice so ozko povezane s ScrumBut-i:	Določitev glavnega lastnika izdelka, ki koordinira ostale lastnike izdelkov, ki delajo z različnimi razvojnimi ekipami. Ta vloga skrbi za urejen seznam uporabniških zgodb in jasno komunikacijo z ostalimi lastniki izdelka ter ostalimi interesnimi skupinami (npr. s stranko). Določitev samo enega lastnika izdelka. V primeru, ko ena oseba ne more opraviti dela, razmislite o sledečih predlogih: • omejitev števila podrobnosti v uporabniških zgodbah, saj so mogoče te preveč podrobno napisane; • omejitev načrtovanja prihajajočih iteracij na največ dve ali tri iteracije vnaprej, saj bolj podrobno načrtovanje najverjetneje ne bo prineslo dodane vrednosti (na ta način omejite število uporabniških zgodb v seznamu uporabniških zgodb); • zmanjšajte število podrobnosti, s katerimi je lastnik izdelka soočen tako, da prenesete del odgovornosti na razvojno ekipo; • zaposlite primernega lastnika izdelka z vsemi potrebnimi znanji, ki so potrebni za uspešno opravljanje vloge; • pošljite lastnika izdelka na dodatno izobraževanje.

• seznam uporabniških zgodb ni pregleden (SB26), • neustrezni elementi seznama uporabniških zgodb niso odstranjeni (SB27) in • seznam uporabniških zgodb ni urejen (SB28). Nejasna odgovornost in lastništvo, kar lahko povzroči, da nihče ni odgovoren za kakovost končnega produkta. Ping-Pong igra krivde: Lastniki izdelkov preusmerijo težave na druge ekipe in s tem ne dosežejo optimalnega razvoja, ki prinese največ dodane vrednoti za stranko. Počasnost pri doseganju rezultatov zaradi velikih odvisnosti med ekipami, ki jih vodijo različni lastniki izdelkov. Počasne odločitve, saj se morajo lastniki izdelkov uskladiti, preden pridejo do skupnega kompromisa.	

4.2 SB2: LASTNIK IZDELKA NI ODGOVOREN ZA UREJANJE UPORABNIŠKIH ZGODB V SEZNAMU UPORABNIŠKIH ZGODB

Tabela 8: Posledice in predlogi za izboljšavo SB2

Možne posledice	**Predlogi za izboljšavo**
Lastnik izdelka nima avtoritete za odločanje, katere naloge bodo razvite in katere ovržene. Posledično naloge v seznamu uporabniških zgodb niso razvrščene. To privede do tega, da so pomembne funkcionalnosti spregledane in da se razvijejo nekatere nepotrebne funkcionalnosti, ki ne prinesejo dodane vrednosti za stranko. Pri takšnem načinu dela obstaja tudi velika verjetnost, da se razvijejo vse uporabniške zgodbe iz seznama uporabniških zgodb. Omenjene posledice so ozko povezane s ScrumBut-i: • seznam uporabniških zgodb ni pregleden (SB26), • neustrezni elementi seznama uporabniških zgodb niso odstranjeni (SB27), • seznam uporabniških zgodb ni urejen (SB28).	Takšna uporaba vloge lastnika izdelka se pogosto zgodi, ko Scrum še ni popolnoma sprejet v podjetju in vse interesne osebe ne vidijo dodane vrednosti uporabe Scrum-a. Poskusite organizirati izobraževanje za vodstvo, kjer bi le-ti pridobili pregled prednosti, če bi vlogo lastnika izdelka ustrezno uporabljali.

Lastnik izdelka izgubi motivacijo za delo in interes do uporabe Scrum ogrodja. Člani ekipe izgubijo interes do uporabe Scrum ogrodja in ne vidijo dodane vrednosti pri vlogi lastnika izdelka. Odgovornost za razvoj izdelkov in sprejemanja odločitev je razdrobljena po podjetju, kar povzroči nejasne komunikacijske kanale in počasne odločitve pri razvoju produkta (predvsem v večjih podjetjih). Drugi člani v podjetju so v neposrednem stiku s stranko (npr. vodstvo).	

4.3 SB3: V EKIPI NI LASTNIKA IZDELKA

Tabela 9: Posledice in predlogi za izboljšavo SB3

Možne posledice	**Predlogi za izboljšavo**
Seznam uporabniških zgodb ni definiran ali ni dovolj podroben. Posledično nastane velika verjetnost, da vizija produkta ni jasna in seznam uporabniških zgodb ne odraža dejanskih zahtev od stranke. Zaradi tega razvojna ekipa razvije produkt po njihovi intuiciji, kar lahko vodi do tega, da se razvijejo funkcionalnosti, ki imajo majhno vrednost za stranko in bi sicer bile na dnu seznama uporabniških zgodb ali bi bile popolnoma prezrte. To lahko pripelje do spreminjanja že izdelanih funkcionalnosti in veliko dodatnega dela, ko pridobijo povratne informacije od stranke. Le-to ustvari razočaranje, frustracijo, negativen vpliv na proračun, zamujene roke izdaje, slabšo kakovost produkta in manjšo dodano vrednost končnega produkta. Omenjene posledice so ozko povezane s ScrumButi: • seznam uporabniških zgodb ni pregleden (SB26), • neustrezni elementi seznama uporabniških zgodb niso odstranjeni (SB27), • seznam uporabniških zgodb ni urejen (SB28), • seznam uporabniških zgodb ne obstaja	Vloga lastnika izdelka je obvezna in brez nje bo Scrum ekipa čutila določene že prej omenjene posledice. Tukaj lahko še omenimo določene prednosti, če vlogo lastnika izdelka uporabimo za: • olajšanje komunikacije med interesnimi skupinami, • povečanje funkcionalnosti produkta z upoštevanjem najboljše donosnosti naložbe, • podpiranje dolgoročnega in kratkoročnega načrtovanja razvoja produkta, • uporabo dobro definiranega seznama uporabniških zgodb za povečanje učinkovitosti razvoja, • določanje prednostnih nalog in ohranjanje osredotočenosti vseh članov ekipe.

(SB29). Vlogo lastnika izdelka prevzame projektni vodja, ki ni toliko osredotočen na povečanje donosnosti naložb in Scrum procese ne uporablja optimalno zaradi pomanjkanja znanja. To povzroči, da je uporabna vrednost Scrum-a manjša kot sicer. Vloga projektnega vodje se osredotoča na čim hitrejšo izvedbo znotraj finančnih okvirjev, kar lahko vpliva na slabšo kakovost končnega produkta in večjo verjetnost za razočaranje s strani stranke. Razvojna ekipa prevzame odgovornosti lastnika izdelka. To povzroči, da razvojna ekipa posveti več časa razjasnitvi funkcionalnih vidikov in produktnih zahtev, namesto da se osredotočajo na razvoj produkta. Nepotrebna prerazporeditev časa in energije se neposredno negativno odraža na proračunu, rokih izdaje produkta in dodani vrednosti za stranko. Vodstvo prevzame vlogo lastnika izdelka. To pogosto vodi do zamud, saj vodstvo nima časa, da bi se posvetili razvoju produkta (značilno za manjša podjetja).	

4.4 SB4: V EKIPI NI SKRBNIKA SCRUM PROJEKTA

Tabela 10: Posledice in predlogi za izboljšavo SB4

Možne posledice	**Predlogi za izboljšavo**
Scrum ekipa sčasoma vse bolj in bolj odstopa od okvirjev pravilne uporabe Scrum procesov in metod. To posledično zmanjša učinkovitost Scrum ekipe ter prinese veliko različnih težav, ki so povezane z različnimi posledicami, ki so omenjeni pri ostalih ScrumBut-ih. Nihče v ekipi ne odstranjuje ovir, ki so na poti do učinkovitega razvoja. Takšne ovire so lahko povezane s procesi v podjetju, z odnosi v razvojni ekipi, z razvojnim okoljem, s pokvarjeno strojno opremo, z odsotnostjo lastnika izdelka itd. Ker se število ovir ne zmanjšuje (temveč povečuje), Scrum ekipa ne poveča svoje učinkovitosti iz iteracijo v iteracijo in ne doseže optimalne	Vloga skrbnika Scrum projekta je obvezna za Scrum ekipe, ki so začele uporabljati Scrum in v ekipah, kjer Scrum procesi še vedno niso popolnoma ustaljeni. Vloga skrbnika Scrum projekta je zelo priporočljiva tudi v ekipah, kjer člani ekipe uporabljajo Scrum že dlje časa in jim je delo s Scrum ogrodjem navada. Brez skrbnika Scrum projekta ekipa verjetno ne bo delovala najbolj učinkovito, saj se ne bomo znebili vseh preprek, ki nas ovirajo pri učinkovitem razvoju. Tukaj lahko še omenimo določene prednosti, če vlogo skrbnika Scrum projekta uporabimo: • reševanje motenj in ovir, da se lahko preostali del ekipe osredotoči na delo, ki bo prineslo želeni rezultat, • vodenje uvedbe Scrum-a in učenje drugih

strukture, katero bi bilo mogoče doseči s skrbnikom Scrum projekta.	članov, kako pravilno uporabljati Scrum procese in metode, • izvajanje sprememb za povečanje produktivnosti ekipe, • vzpodbujanje samoorganizacije razvojne ekipe, • sodelovanje z drugimi skrbniki Scrum projekta za izboljšanje učinkovitosti metodologije v podjetju.

4.5 SB5: EDEN LASTNIK IZDELKA V RAZLIČNIH EKIPAH

Tabela 11: Posledice in predlogi za izboljšavo SB5

Možne posledice	**Predlogi za izboljšavo**
Lastnik izdelka ima premalo časa za bistvene zadeve pri razvoju produkta, kot so urejanje seznama uporabniških zgodb, kratko in dolgoročno načrtovanje vizije produkta, razumevanje stranke in njihovih zahtev, pridobivanje najvišje donosne vrednosti produkta itd. To prinese veliko različnih težav za razvojne ekipe in tako razvojni procesi ne dosežejo optimalne ravni. Delo lastnika izdelka zahteva veliko časa, saj je v kontaktu z veliko različnimi interesnimi skupinami. Med drugim mora lastnik izdelka v takšni vlogi pogosto menjati kontekst pogovora, saj lahko dela na različnih produktih, kar ni najbolj učinkovito in je lahko tudi zelo stresno. Posledice tega ScrumBut-a so ozko povezane s ScrumBut-i: • seznam uporabniških zgodb ni pregleden (SB26), • neustrezni elementi seznama uporabniških zgodb niso odstranjeni (SB27), • seznam uporabniških zgodb ni urejen (SB28).	Scrum zahteva, da ima vsaka Scrum ekipa svojega lastnika izdelka, saj se drugače zgodijo prej omenjene posledice. V primeru, da podjetje začasno ne more pridobiti dodatnega lastnika izdelka, lahko poizkusite poenostaviti delo lastnik izdelka, ki deluje v več različnih Scrum ekipah s sledečimi predlogi: • Udeležba le na bistvenih sestankih. Za lastnika izdelka sta najpomembnejša sestanka za izboljšavo seznama uporabniških zgodb in sestanek za predstavitev rezultatov iteracije. Ostali sestanki iz Scrum strukture so lahko poljubni za udeležbo lastnika izdelka. Med drugim mora lastnik izdelka premisliti, če se mora udeležiti tudi preostalih sestankov izven Scrum strukture, kot so sestanki s prodajnim oddelkom, tehnični sestanki s sistemskimi arhitekti itd. • Zmanjšanje števila podrobnosti v uporabniških zgodbah. Pri temu mora biti lastnik izdelka previden, da razvojni ekipi predstavi zadosti podrobnosti, da razumejo, kaj je cilj razvoja funkcionalnosti in cilj celotnega razvoja na produktu. • Posvetovanje z razvojno ekipo, če lahko prevzamejo določene dele pri opredelitvi uporabniških zgodb. Na primer razdelitev uporabniških zgodb na manjše uporabniške zgodbe, opredelitev podrobnosti v uporabniških zgodbah itd.

4.6 SB6: LASTNIK IZDELKA DELA ZA STRANKO

Tabela 12: Posledice in predlogi za izboljšavo SB6

Možne posledice	**Predlogi za izboljšavo**
Lastnik izdelka od stranke je lahko preveč zaseden in ne uspe pripraviti seznama uporabniških zgodb za razvojno ekipo. Ekipi tako niso jasne zahteve za razvoj produkta. Med strankinim lastnikom izdelka in drugimi člani Scrum ekipe lahko dodatno pride do otežene komunikacije zaradi različnih organizacijskih ovir med podjetji. Strankin lastnik produkta ne razume dinamike dela razvojne ekipe ter lahko razvijalcem vsiljuje, kako morajo razviti določeno funkcionalnost, ali spreminja vsebino seznama nalog iteracije. Takšna dejanja zmotijo delo razvojne ekipe, kar zmanjša učinkovitost dela in povzroči manjšo zavezanost razvojne ekipe za doseganje cilja iteracije. Posledice tega ScrumBut-a so ozko povezane s posledicami ScrumBut-a: vodstvo spreminja seznam nalog iteracije med potekom iteracije (SB23).	V primeru, ko lastnik izdelka dela za stranko poskusite: • na temelju prej omenjenih posledic prepričati stranko, da je za razvoj produkta bolj učinkovito imeti svojega lastnika izdelka; • narediti ekipo lastnikov izdelkov v vašem podjetju, kjer sodeluje tudi strankin lastnik izdelka.

4.7 SB7: PRESEGANJE MEJ VELIKOSTI EKIPE

Tabela 13: Posledice in predlogi za izboljšavo SB7

Možne posledice	**Predlogi za izboljšavo**
Podaljševanje trajanja dnevnega sestanka. V ekipi je morda preveč članov, če so dnevni sestanki učinkoviti, ampak še vedno trajajo več kot 15 minut. V primeru, ko preveč članov deli informacije z drugimi, je težko ostati zbran in vse podatke sprejeti, saj smo ljudje omejeni, koliko informacij lahko sprejmemo. To povzroči, da se člani ekipe zaprejo, razmišljajo le o svojem napredku in ne iščejo priložnosti, kako bi lahko pomagali drugim članom ekipe. Več podrobnosti o posledicah lahko najdete pri ScrumBut-u: Podaljševanje trajanja dnevnega sestanka (SB20). Organiziranje več dnevnih sestankov z določenimi člani ekipe. Pri velikih Scrum ekipah lahko pride do podaljševanja dnevnih sestankov, kar privede do tega, da se dnevni sestanki zgodijo v ločenih	Razdelite večje ekipe na dve manjši ekipi. Pri tem poskrbite, da ima vsaka ekipa podobno izkušene ljudi in določite cilje novih ekip. Pri delitvi je potrebno upoštevati mnenja članov ekipe, saj oni najbolje poznajo dinamiko ljudi v ekipi in vedo, s katerimi člani bi nove ekipe najboljše delovale. Potrebno je tudi upoštevati, da so novonastale razvojne ekipe večfunkcijske, saj mora biti vsaka razvojna ekipa sposobna samostojno razviti nove funkcionalnosti na produktu. Za usklajevanje ekip je smiselno preučiti metode skaliranja (ang. scaling methods), kot so: • Scrum of Scrums (SoS), • Large Scale Scrum (LeSS), • Scaled Agile Framework (SAFe), • Scrum@Scale.

manjših ekipah. Podrobnejše posledice tega vzorca so opisane v ScrumBut-u: Več dnevnih sestankov z različnimi udeleženci (SB17). Podaljševanje sestanka za izboljšavo seznama uporabniških zgodb in sestanka za načrtovanje iteracije. Več članov v ekipi lahko pomeni, da lahko pridemo do boljše odločitve, kako določeno funkcionalnost razviti. Preveč članov v ekipi pa povzroči, da porabimo preveč časa za deljenje vseh mnenj in le-to zmanjša učinkovitost razvojne ekipe. Zelo podrobni pogovori pogosto povzročajo podaljševanje sestankov, kar povečuje nezadovoljstvo razvojne ekipe. Če opazite, da člani ekipe težko najdejo skupno razumevanje, to lahko pomeni, da je razvojna ekipa prevelika in jo je potrebno razdeliti. Podrobnejše posledice so opisane v ScrumBut-u: Podaljšanje trajanja sestanka za načrtovanje iteracije (SB22).	

4.8 SB8: DRUGI NAZIVI SO UPORABLJENI PRI RAZVOJNI EKIPI

Tabela 14: Posledice in predlogi za izboljšavo SB8

Možne posledice	**Predlogi za izboljšavo**
Člani razvojne ekipe niso enakopravni. Določeni člani razvojne ekipe prevzamejo večjo odgovornost, kar povzroči, da drugi člani razvojne ekipe ne potrebujejo prevzeti odgovornosti. Zaradi tega se vsi člani razvojne ekipe ne trudijo po najboljših močeh, da dosežejo zastavljeni cilj.	Uporabljanje drugih nazivov v razvojni ekipe razdeli ekipo na več nivojev in porazdeli odgovornost samo na določene osebe. Predlagamo, da odstranite vse dodatne nazive pri razvijalcih v razvojni ekipi. S tem boste dosegli skladnost ekipe in lažje dosegli zastavljene cilje.

4.9 SB9: RAZVOJNA EKIPA NI VEČ-FUNKCIJSKA

Tabela 15: Posledice in predlogi za izboljšavo SB9

Možne posledice	**Predlogi za izboljšavo**
Razvojna ekipa je zmožna razviti le določen del funkcionalnosti in s tem ne more povečati uporabne vrednosti za stranko po koncu iteracije. Razvojna ekipa je zmožna razviti funkcionalnosti, toda le-te niso testirane takoj, saj podjetje uporablja ločeno testno ekipo. Takšen način dela povzroča počasnejše povratne informacije med testno in	Scrum ogrodje je narejeno tako, da stranka po vsaki iteraciji pridobi nove funkcionalnosti na produktu. V primeru, da razvojna ekipa ni večfunkcijska, tega ne moremo vedno omogočiti. Tudi v primeru, ko Scrum uporabljamo v okolju, kjer povečanje funkcionalnosti na produktu ni možno v vsaki iteraciji (npr. razvoj strojne opreme), je večfunkcijska ekipa potrebna za hiter in učinkovit razvoj. Drugače pogosto pride do

razvojno ekipo ter tudi več motenj pri razvoju za razvojno ekipo. Razvijalci morajo pogosto takoj prenehati z delom na trenutni funkcionalnosti in rešiti napako na prejšnji, saj lahko na ta način prej podajo popravljeno funkcionalnost preizkuševalcu, da jo le-ta lahko ponovno preizkusi.	nepredvidenih zamud. Večfunkcijska ekipa ne pomeni, da mora imeti vsak član razvojne ekipe vsa potrebna znanja iz vseh področij. Razvojna ekipa kot skupina mora imeti skupek znanj, da sami razvijejo in preizkusijo nove funkcionalnosti na produktu, preden so predane stranki. Tukaj lahko vidite še dodatne prednosti večfunkcijske ekipe: • izboljšana uskladitev med različnimi funkcionalnimi področji, • povečanje inovacije v produktih in procesih, • skrajšan čas razvojnega cikla ter s tem hitrejše povratne informacije od interesnih skupin (npr. stranke).

4.10 SB10: DNEVNI SESTANKI SO PREKLICANI ALI PRELOŽENI

Tabela 16: Posledice in predlogi za izboljšavo SB10

Možne posledice	**Predlogi za izboljšavo**
Obstaja velika nevarnost, da se člani ekipe ne uskladijo glede trenutnega dela. Zaradi tega ne uspejo maksimalno povečati učinkovitosti za doseganje cilja iteracije. Pri temu ne odkrijejo težav sodelavcev, ne morejo z njimi deliti svoja mnenja in nuditi pomoč, da se težava prej razreši. Določene odločitve se zgodijo kasneje, saj razvijalci in ostali člani Scrum ekipe ne pridobijo informacij o trenutnem razvoju. Razvijalci ne povečujejo svojega znanja tako hitro, saj sodelavci medsebojno redkeje delijo svoja mnenja in izkušnje.	Če člani ekipe zamujajo, jih ne čakajte. Če se nekdo ne more udeležiti sestanka, ga ne prekličite ali preložite. Ne zmanjšujte pogostosti sestanka in ga izvajajte dnevno. Spreminjanje pogostosti dovolite le, če imate opravka z zelo izkušeno Scrum ekipo, ki se strinja, da se pogostost sestanka zmanjša (predvsem v primeru, ko je ekipa manjša in se člani ekipe lahko medsebojno uskladijo). Če se pogostost sestanka zmanjša, mora to biti zadnja stvar, ki jo spremenite glede dnevnih sestankov. Najprej preverite, ali obstajajo drugi razlogi, da člani Scrum ekipe ne vidijo prednosti v dnevnih sestankih. Morate se osredotočiti na podrobno preučitev vzrokov in jih razrešiti. Večino težav z dnevnim sestankom povzroča nepravilna izvedba. Več posledic in rešitev glede dnevnih sestankov lahko vidite pri ScrumBut-ih: • Nekateri razvijalci se ne udeležujejo dnevnih sestankov (SB19). • Podaljševanje trajanja dnevnega sestanka (SB20).

4.11 SB11: RETROSPEKTIVNI SESTANKI ITERACIJE SO PREKLICANI, PRELOŽENI ALI NISO NAČRTOVANI

Tabela 17: Posledice in predlogi za izboljšavo SB11

Možne posledice	**Predlogi za izboljšavo**
Člani razvojne ekipe ne odkrijejo potencialne izboljšave pri razvoju. Med seboj si ne razdelijo nalog za izboljšavo in s tem ovirajo kontinuiran napredek. S takšnim pristopom razvojna ekipa ne doseže večje učinkovitosti v naslednjih iteracijah. Konflikti v Scrum ekipi se lahko povečajo, saj člani ne govorijo o težavah s sodelavci in le-te ne rešujejo.	Retrospektivni sestanek iteracije je potreben, če želi Scrum ekipa povečati svojo učinkovitost pri delu. To morajo razumeti vsi vpleteni pri razvoju produkta. Predvsem je pomembno, da to razume vodstvo, ki ne sme ovirati retrospektivne sestanke zaradi časovnega pritiska. Z neizvajanjem retrospektivnega sestanka učinkovitost Scrum ekipe dolgoročno ne doseže svojega potenciala, kar vpliva na končni produkt in zadovoljstvo stranke. Sestanka se morajo udeležiti vsi člani Scrum ekipe. Člani razvojne ekipe, lastnik izdelka in skrbnik Scrum projekta. Interesne skupine ali vodstvo se lahko pridružijo sestanku samo takrat, ko jih povabi Scrum ekipa. V nasprotnem primeru se lahko zgodi, da se člani Scrum ekipe zadržujejo pri izražanju mnenj in tako ne odkrijemo bistvenih izboljšav za doseganje kontinuiranega napredka. V primeru, ko se eden izmed članov Scrum ekipe ne more udeležiti sestanka zaradi odsotnosti (v večini primerov je to lastnik izdelka), poskusimo najti način, da je to vedno možno. Na primer uporaba video konferenčnih aplikacij, ko je lastnik izdelka na poslovnem potovanju. V primeru, ko to ni možno, se sestanek izpelje brez te osebe. Poznamo veliko različnih načinov, kako retrospektiven sestanek voditi, toda najosnovnejša vprašanja, ki si jih moramo postaviti so: • Kaj je delovalo dobro v tej iteraciji? • Kaj ni delovalo dobro v tej iteraciji? • Kaj bomo izboljšali? V primeru, ko imamo občutek, da imamo v ekipi premalo tem za pogovor, lahko poskusimo uporabiti različne metode za vodenje retrospektivnega sestanka, saj lahko z drugačnim naborom vprašanj odkrijemo težave, ki jih prej nismo opazili. Več različnih metod lahko najdete v knjigi Agile Retrospectives [15]. Člani ekipe lahko dobijo občutek, da se sestanek ponavlja in da je dolgočasen, ko uporabljamo enako metodo dlje časa. Pogosto je

	to vpliv. zakaj ekipe ne pridejo do novih inovativnih idej, kaj lahko pri Scrum ekipi dodatno izboljšamo. Moramo poskusiti najti drugačen način določanja ukrepov in nalog v sestanku, če akcije iz prejšnjih retrospektivnih sestankov pogosto niso opravljene. Neopravljene naloge zmanjšujejo razumevanje bistva tega sestanka in zaradi tega se člani ekipe poskusijo izogniti retrospektivnemu sestanku. Predlagamo, da se za vsako nalogo določi ena oseba, ki bo najverjetneje nalogo lahko opravila do naslednjega retrospektivnega sestanka. Prepiranje v retrospektivnem sestanku iteracije ni nič nepričakovanega. Pomembno je prepire usmerjati v smeri reševanja težav in ne v smeri obsojanja sodelavcev. Tukaj mora skrbnik Scrum projekta pomagati ekipi najti pravilni način izvedbe sestanka. Na ta način se skozi iteracije izboljšuje timsko delo Scrum ekipe in postopoma rešujemo težave, na katere naletimo. V primeru obsojanja in nepravilne moderacije retrospektivnega sestanka bodo člani dobili odpor do sestanka, se ga ne bodo želeli več udeleževati, kar vodi do preklica ali preložitve retrospektivnega sestanka.

4.12 SB12: SESTANKI ZA PREDSTAVITEV REZULTATOV ITERACIJE SO PREKLICANI ALI PRELOŽENI

Tabela 18: Posledice in predlogi za izboljšavo SB12

Možne posledice	**Predlogi za izboljšavo**
Od pomembnih interesnih skupin (npr. stranka, sistemski arhitekt …) ne pridobimo povratnih informacij pravočasno. To lahko povzroči: • Veliko kasnejših predelav produkta, kar pomeni dodatno delo za razvojno ekipo in zamude pri projektih. Med drugim lahko to povzroči nižjo kakovost produkta (npr. že zastavljena arhitektura produkta ne omogoča razvoja zamišljene funkcionalnosti, sprememba arhitekture pa zahteva preveč časa). • Slabši nabor funkcionalnosti, saj nismo pravočasno pridobili zanimivih idej, da jih upoštevamo pri razvoju produkta.	V določenih primerih, ko ima Scrum ekipa zelo dober odnos s stranko in ostalimi ključnimi interesnimi skupinami, se zdi sestanek za predstavitev rezultatov iteracije nepotreben. Pogosto se zgodita dva scenarija: • Stranka in ostale ključne interesne skupine so zelo pogosto v kontaktu z lastnikom izdelka in z razvojno ekipo. To lahko povzroči več motenj pri delu za člane razvojne ekipe in zmanjša njeno učinkovitost. Posledično dobi stranka končni produkt slabše kakovosti ali ga dobi kasneje, kot je bilo pričakovano. Predlagamo, da se kontakti s stranko omejijo na določene Scrum sestanke, saj na ta način

	ne zmanjšamo učinkovitosti ekipe, kljub temu pa imamo pogost kontakt in dober odnos s stranko in ostalimi ključnimi interesnimi skupinami. • Stranka in ostale ključne interesne skupine so zelo pogosto v kontaktu z lastnikom izdelka, ampak niso v kontaktu z razvojno ekipo. V primeru, ko ne izvedemo sestanka za predstavitev rezultatov iteracije, razvojni ekipi primanjkuje neposredna komunikacija s stranko in ključnimi interesnimi skupinami. Zaradi tega obstaja nevarnost, da razvojna ekipa ne razume točnih želja od stranke in ne pridobijo koristnih nasvetov od ključnih interesnih skupin. Lastnik izdelka ne more predati želje tako kakovostno, kot lahko to storimo z neposredno komunikacijo. Zato predlagamo, da vedno izvedemo sestanek za predstavitev rezultatov iteracije, kjer se pridružijo tudi stranka in ključne interesne skupine. V primeru, ko ne dosežemo cilja iteracije, ne prekličemo ali preložimo sestanka za predstavitev rezultatov iteracije, saj lahko še vedno pridobimo bistvene povratne informacije za nadaljnji razvoj. Cilj sestanka je pridobiti povratne informacije, s katerimi lahko poskrbimo za kontinuirani napredek na produktu. Najbolj bistveno je pridobiti povratne informacije od oseb, s katerimi se ne srečujemo pogosto, saj nam lahko podajo nov pogled na funkcionalnost ali produkt. Te informacije upoštevamo v naslednjem sestanku za načrtovanje iteracije in sestanku za izboljšavo seznama uporabniških zgodb. Sestanka za predstavitev rezultatov iteracije nikoli ne prekličemo ali preložimo. Lahko ga pa skrajšamo tako, da predstavimo samo funkcionalnosti, za katere mislimo, da bomo dobili uporabne povratne informacije od udeležencev sestanka. Pri tem je potrebno proaktivno spremljanje, kdo se bo sestanka udeležil in posledično spreminjanje seznama funkcionalnosti, ki jih bomo predstavili. Na ta način lahko skrajšamo potreben čas za pripravo na sestanek in tudi trajanje sestanka.

	Čas priprave na sestanek je lahko časovno potraten, kar povzroči, da se ekipe odločijo, da sestanka ne bodo izvedle. Da se temu izognemo, lahko izboljšamo pripravljalni čas za sestanek tako, da: Poskušamo najti lahek pristop, kjer prikažemo dokumentacijo, ki je napisana sočasno ob razvoju produkta. Predvsem se osredotočimo na dele z vizualizacijo, ki ostalim udeležencem lažje predstavi delovanje funkcionalnosti. Prikažemo delujoči produkt, ki je rezultat dela v iteraciji. Če določen del produkta ne deluje, ga ne pripravljamo za namen predstavitve, saj to povzroča dodatno delo, ki se ne odraža na končnem produktu.

4.13 SB13: SESTANKI ZA NAČRTOVANJE ITERACIJE SO PREKLICANI ALI PRELOŽENI

Tabela 19: Posledice in predlogi za izboljšavo SB13

Možne posledice	**Predlogi za izboljšavo**
Zaradi odsotnosti člana Scrum ekipe (najpogosteje lastnika izdelka) je sestanek za načrtovanje iteracije prestavljen. Posledično na začetku iteracije njen cilj ni jasen in razvojna ekipa ne ve, na kaj mora biti osredotočena. Le-to se pozna pri manjši motivaciji, kar vpliva na manjšo učinkovitost razvojne ekipe. Med drugim lahko člani ekipe začnejo delo na funkcionalnostih, ki sicer ne bi bile dodane v trenutno iteracijo. Razvojna ekipa dokončuje naloge iz prejšnje iteracije, saj niso pravočasno dosegli cilja iteracije, zato so sestanek za načrtovanje iteracije preložili. Posledično bodo razvijalci imeli občutek, da se iteracije lahko podaljšujejo, kar vpliva na manjšo zavzetost za pravočasno doseganje ciljev iteracij.	V primeru, ko lastnik izdelka nima časa za udeležitev sestanka za načrtovanje iteracije (dopust, poslovno potovanje …) mora lastnik izdelka: • poskusiti najti čas, da se sestanka udeleži s pomočjo video konferenčnih aplikacij, • zagotoviti primerno zamenjavo, ki poda vse strankine želje in njegovo vizijo glede cilja naslednje iteracije. Preklic ali preložitev sestanka za načrtovanje iteracije ni priporočljiv, saj lahko povzroči prej omenjene posledice. V primeru, ko prejšnje iteracije ni bilo možno dokončati (cilj iteracije ni bil dosežen), je predlagano, da nedokončane uporabniške zgodbe vrnemo v seznam uporabniških zgodb. V sestanku za načrtovanje iteracije se nato odločimo, ali jih damo v seznam nalog trenutne iteracije ali ne. V določenih primerih lahko pride do spremembe zahtev od stranke, kar povzroči, da imajo druge uporabniške zgodbe višjo prioriteto od nedokončanih funkcionalnosti iz prejšnje iteracije.

4.14 SB15: EKIPA NE UPORABLJA ITERACIJ

Tabela 20: Posledice in predlogi za izboljšavo SB15

Možne posledice	**Predlogi za izboljšavo**
Izguba ritma in dela na cilju iteracije vpliva na slabšo učinkovitost razvojne ekipe. Lastnik izdelka mora biti vedno na voljo razvojni ekipi, kar povzroča veliko stresa za lastnika izdelka. Takojšnjo dosegljivost lastnika izdelka je tudi zelo težko doseči, kar lahko vpliva, da razvojna ekipa ne pridobi potrebne informacije, ko jih potrebujejo.	Po želji po boljši odzivnosti za strankine želje določene ekipe ne uporabljajo iteracij. S takšno strukturo ima lahko razvojna ekipa veliko težav s stresom in učinkovitostjo. V temu primeru je delo razvojne ekipe vedno moteno, saj stranka narekuje, na katerih funkcionalnostih morajo delati in razvojna ekipa se mora hitro odzvati. Scrum ekipa tako ne more pripraviti strateškega načrta, kako razviti produkt, kar najbolj vpliva na kakovost produkta in pogosto tudi na podaljšanje projekta. Veliko število ekip, ki so izkusile takšno delo, je ponovno začelo uporabljati iteracije, saj takšno delo ni prineslo želene dodane vrednosti. Predlagamo vam, da vedno uporabljate iteracije, stranki pa jasno razložite prednosti tega načina dela. V primeru, ko mora stranka imeti zaradi narave dela hitro odzivnost Scrum ekipe, predlagamo, da skrajšate trajanje iteracije na čas, ki bo omogočal takšno delo s stranko. Določeni projekti lahko trajajo zelo kratko, kar lahko vpliva, da ekipe ne uporabljajo iteracij za razvoj. V tem primeru bi si na vašem mestu postavili vprašanje, ali je Scrum najboljša izbira za kratke projekte? Namreč takšni projekti so po navadi manj kompleksni in Scrum lahko prinese preveč dodatnega usklajevanje, ki se ne odraža pri uspešnosti izvedbe končnega produkta.

4.15 SB16: ZDRUŽEVANJE SESTANKA ZA PREDSTAVITEV REZULTATOV ITERACIJE IN RETROSPEKTIVNEGA SESTANKA ITERACIJE

Tabela 21: Posledice in predlogi za izboljšavo SB16

Možne posledice	**Predlogi za izboljšavo**
Združevanje sestanka nakazuje nerazumevanje namena posameznega sestanka. To lahko pomeni, da se Scrum ekipa v sestankih ne osredotoča na pravilna vprašanja, ki jih morajo postaviti, da pridejo do pričakovanega cilja. S takšnim	Ekipe, ki združujejo ali želijo združiti sestanek za predstavitev rezultatov iteracije in retrospektivni sestanek iteracije, se morajo poglobiti v to, kaj je namen posameznega sestanka. Sestanek za predstavitev rezultatov iteracije vključuje metodo

pristopom Scrum ekipe popolnoma ne izkoristijo uporabne vrednosti Scrum ogrodja.	pregleda in prilagajanja z osredotočenostjo na produkt, dokler retrospektivni sestanek iteracije vključuje metodo pregleda in prilagajanja z osredotočenostjo na proces. Sestanka za predstavitev rezultata iteracije se udeležijo člani Scrum ekipe in tudi vse interesne skupine. Pri retrospektivnem sestanku se večinoma udeležijo le člani Scrum ekipe, saj drugače določeni člani Scrum ekipe ne bi odkrito delili svoja mnenja. Tudi če ločena sestanka trajata dlje časa, je zelo priporočljivo imeti sestanka ločena. Le na ta način lahko najbolje izkoristimo namen Scrum ogrodja.

4.16 SB17: VEČ DNEVNIH SESTANKOV Z RAZLIČNIMI UDELEŽENCI

Tabela 22: Posledice in predlogi za izboljšavo SB17

Možne posledice	**Predlogi za izboljšavo**
Neusklajenost razvojne ekipe. Manjša možnost uporabe sinergij med člani razvojne ekipe. Manjša verjetnost doseganja cilja iteracije.	Takšen vzorec pogosto opazimo, ko Scrum ekipa presega priporočeno število članov. Posledice in predloge za izboljšavo pri delu z velikimi ekipami lahko opazite pri ScrumBut-u: Preseganje mej velikosti ekipe (SB7). Vzorec več dnevnih sestankov z različnimi udeleženci opazimo tudi pri delu s porazdeljenimi ekipami. Pri takšnem načinu dela dodatno težavo predstavlja časovna razlika, zaradi katere imajo razvojne ekipe na različnih lokacijah ločene dnevne sestanke. To predstavlja veliko nevarnost, da Scrum ekipa ne doseže svojega potenciala, saj je komunikacija že tako med porazdeljenima ekipama slabša. V primeru velike časovne razlike poskusite organizirati dnevni sestanek tako, da se lahko vse ekipe udeležijo sestanka (tudi če to pomeni prilagajanje s strani ene ali več ekip). Na ta način bomo poskrbeli za boljšo usklajenost med ekipami in večjo verjetnost za doseganje cilja iteracije.

4.17 SB18: SESTANEK ZA PREDSTAVITEV REZULTATOV ITERACIJE POTEKA BREZ KLJUČNIH INTERESNIH SKUPIN

Tabela 23: Posledice in predlogi za izboljšavo SB18

Možne posledice	**Predlogi za izboljšavo**
Enake posledice kot pri ScrumBut-u: Sestanki za predstavitev rezultatov iteracije so preklicani ali preloženi (SB12).	V določenih primerih so stranka in druge ključne interesne skupine odtujene od Scrum ekipe in tudi ne sodelujejo pogosto na sestanku za predstavitev rezultatov iteracije. To lahko vpliva na posledice, ki smo jih omenili v ScrumBut-u: Sestanki za predstavitev rezultatov iteracije so preklicani ali preloženi (SB12). Predlagamo, da stranki obrazložite prednosti hitrih povratnih informacij ter s tem hitre prilagoditve seznama uporabniških zgodb. Na ta način zagotovimo hitrejši razvoj produkta s funkcionalnostmi, ki predstavljajo največjo uporabno vrednost za stranko.

4.18 SB19: NEKATERI RAZVIJALCI SE NE UDELEŽUJEJO DNEVNIH SESTANKOV

Tabela 24: Posledice in predlogi za izboljšavo SB19

Možne posledice	**Predlogi za izboljšavo**
Ne delimo informacije med vsemi člani razvojne ekipe in tako zmanjšamo verjetnost za najučinkovitejšo doseganje cilja iteracije. Člani, ki se ne udeležijo dnevnega sestanka, imajo lahko ključne informacije, ki jih ne delijo z drugimi člani, če se ne udeležijo dnevnega sestanka, kar vpliva na učinkovitost celotne razvojne ekipe. Le-ta ni zmožna doseči maksimalnega povečanja funkcionalnosti na produktu, kot bi bila zmožna, če bi se vsi člani razvojne ekipe udeležili dnevnega sestanka (manjši doprinos za stranko).	Člani razvojne ekipe se morajo udeležiti dnevnih sestankov, saj je sestanek namenjen le njim in brez vseh članov ekipe je sestanek nesmiseln. Člani razvojne ekipe so edini, ki aktivno sodelujejo v sestanku. Lastnik izdelka, skrbnik Scrum projekta in druge interesne skupine lahko sodelujejo kot poslušalci ter ne smejo motiti poteka sestanka. Le-to mora zagotoviti skrbnik Scrum projekta. V razvojni ekipi moramo poskrbeti, da imajo člani zadosti avtonomije in samostojnosti, saj drugače lahko dojemajo dnevni sestanek kot dolgočasen sestanek za poročanje napredka. To je eden izmed razlogov, zakaj se dnevnega sestanku izogibajo. Dodaten razlog za nezadovoljstvo razvojne ekipe z dnevnim sestankom je tudi ta, da se dnevni sestanki pogosto podaljšujejo in ne trajajo le 15 minut. Več podrobnosti povezanih s tem ScrumBut-om si lahko preberete pri ScrumBut-u: Podaljševanje trajanja dnevnega sestanka (SB20). V primeru, ko razvijalci zjutraj zamujajo ali ne

	pridejo na dnevni sestanek, organizirajte sestanek tik pred kosilom. Na ta način zagotovite, da vsi člani ekipe pridejo na sestanek pravočasno. Dodatne posledice tega so: • dnevni sestanek se ne podaljša, saj so člani ekipe motivirani, da sestanek čimprej zaključijo, • razvijalci se lažje spomnijo, na katerih nalogah delajo in s katerimi težavami se soočajo, • razvijalci imajo manj prekinitev, • razvijalci niso prisiljeni priti na delo ob določeni uri, • člani ekipe bodo bolj verjetno šli skupaj na kosilo, kar vzpodbuja dobre odnose v ekipi. Razprave lahko tudi nadaljujejo med kosilom.

4.19 SB20: PODALJŠEVANJE TRAJANJA DNEVNEGA SESTANKA

Tabela 25: Posledice in predlogi za izboljšavo SB20

Možne posledice	**Predlogi za izboljšavo**
Izguba pozornosti med dnevnim sestankom, kar povzroči, da člani razvojne ekipe ne sodelujejo aktivno pri sestanku. Nezadovoljstvo članov razvojne ekipe. Upiranje dnevnim sestankom in s tem posledično želja razvijalcev: • po preklicu sestanka v določenih dnevih v tednu (Dnevni sestanki so preklicani ali preloženi (SB10)), • da niso prisotni na sestanku (Nekateri razvijalci se ne udeležujejo dnevnih sestankov (SB19)).	Da dosežemo hiter začetek sestanka, lahko določimo osebo, ki bo prva začela. Nato pa nadaljujemo z drugimi osebami v smeri urinega kazalca. Na ta način vsi vedo vrstni red govora brez kakšnega koli posredovanja. Prav tako je dobro, da vedno sestanek začne druga oseba. Nepravilna izvedba dnevnega sestanka je najpogostejši razlog za podaljševanje sestanka. Za povečanje verjetnosti, da sestanek boljše izvedete, predlagamo, da si člani razvojne ekipe postavijo tri vprašanja in na le-ta poskusijo odgovoriti med sestankom: 1. Kaj sem naredil včeraj, da sem pomagal razvojni ekipi doseči cilj iteracije? 2. Kaj bom naredil danes, da bom pomagal razvojni ekipi doseči cilj iteracije? 3. Ali vidim kakšne ovire, ki mene ali razvojno ekipo preprečuje, da dosežemo cilj iteracije? Toda moramo biti pozorni, da sledenje le tem vprašanjem ne privede do nefunkcionalnih sestankov. Številne izkušene ekipe ne sledijo tako strogi strukturi. Pri teh vprašanjih predlagamo, da se ekipe osredotočajo bolj na prihodnje delo kot že narejeno

	delo. Dodatno moramo biti pozorni in se zavedati, da: • dnevni sestanek ni sestanek za poročanje statusa, • ne delimo površnih, nepomembnih informacij v dnevnem sestanku, • podrobne razprave začnemo le, če jih lahko hitro zaključimo in če so pomembne za celotno razvojno ekipo. Podrobne razprave se praviloma zgodijo po sestanku v manjših skupinah, kjer se le-teh udeležijo vsi zainteresirani člani ekipe. Končni cilj srečanja je povečanje učinkovitosti razvojne ekipe pri doseganju zastavljenega cilja. Želimo si, da člani razkrivajo težave, prosijo za pomoč in le-to nudijo, rešujejo probleme, se pogovarjajo in s tem pridobijo koristne informacije. Dnevni sestanki izboljšujejo komunikacijo, pomagajo prepoznati ovire pri razvoju, poudarjajo in vzpodbujajo hitro sprejemanje odločitev ter izboljšajo raven znanja pri razvijalcih. To je ključni sestanek za pregled in prilagoditev, ki je ena od temeljnih načel Scrum-a. Zmanjšanje velikosti ekipe pogosto zmanjša dolžino dnevnega sestanka. Prevelika ekipa povzroči podaljšanje dnevnega sestanka, tudi če je sestanek učinkovito izveden (Preseganje mej velikosti ekipe (SB7)).

4.20 SB21: PODALJŠEVANJE TRAJANJE ITERACIJE MED ITERACIJO

Tabela 26: Posledice in predlogi za izboljšavo SB21

Možne posledice	**Predlogi za izboljšavo**
Razvijalci imajo občutek, da se iteracije lahko podaljšujejo in se ne naučijo, kako naslednjič boljše načrtovati in cilj iteracije doseči pravočasno. To vpliva na manjšo zavzetost razvojne ekipe v prihodnjih iteracijah. Spreminjajoči ritem povzroča, da člani Scrum ekipe in preostale interesne skupine (produktni manager, stranka, sistemski arhitekt …) niso sigurni, kdaj sledijo naslednji Scrum dogodki in zato težje načrtujejo svoj čas. To otežuje prisotnost vseh	Večja negotovost pogosto vodi v podaljševanje iteracij (predvsem, ko ekipa uporablja daljše iteracije). Negotovost pri Scrum ekipah lahko opazimo v različnih oblikah. Recimo neustrezno opredeljene zahteve, razvoj z uporabo nove tehnologije, kompleksen in zelo rizičen razvoj itd. V tem primeru je najboljše, da ekipa začne uporabljati krajše iteracije, saj se s tem lažje nadzoruje potek razvoja. Predlagamo tudi, da si pri uporabi novih tehnologij ekipa vzame čas pred začetkom projekta, da tehnologije preizkusijo in jih tako bolje razumejo.

potrebnih udeležencev na sestankih. V primeru iteracij s fiksno dolžino je lažje, saj vsi udeleženci natančno vedo, kdaj se bo zgodil naslednji Scrum dogodek.	V primeru, da se vzorec podaljševanja iteracij pogosto dogaja zaradi nedokončanih uporabniških zgodb, predlagamo spremembo dolžine iteracije. Razvojna ekipa mora preučiti, zakaj ne uspe dokončati zastavljenih uporabniških zgodb in določiti primerno dolžino iteracije, ki bo to omogočala. Le-to je najboljše storiti med retrospektivnim sestankom iteracije. Pozorni morate biti, da pred spremembo dolžine iteracije preučite tudi druge vplive (velikost uporabniških zgodb, število nalog v seznamu nalog iteracije, motnje pri razvoju …), ki lahko vplivajo na nedoseganje cilja iteracije, saj obstaja možnost, da že uporabljate najbolj primerno dolžino iteracije.

4.21 SB22: PODALJŠANJE TRAJANJA SESTANKA ZA NAČRTOVANJE ITERACIJE

Tabela 27: Posledice in predlogi za izboljšavo SB22

Možne posledice	**Predlogi za izboljšavo**
Nezadovoljstvo članov Scrum ekipe, saj sestanek predolgo traja in se ne morejo osredotočiti na razvoj produkta. Pri dolgih sestankih za načrtovanje iteracije razvijalci izgubijo osredotočenost. Najbolj je to vidno na drugem delu sestanka, ko je potrebno dodati uporabniškim zgodbam podrobnosti, ki jim bodo pomagale pri samem razvoju. V primeru, da razvijalci zaradi utrujenosti in neosredotočenosti ne morejo optimalno sodelovati v razpravi, se lahko zgodi, da se podrobnosti uporabniške zgodbe spremenijo med samim razvojem. Le-to povzroča potrebo po dodatni razpravi in neučinkovitem razvoju.	Z učinkovito pripravo in izvedbo sestanka za načrtovanje iteracije lahko zmanjšamo verjetnost, da se bo sestanek podaljšal. Za dobro pripravo na sestanek je potrebno poskrbeti za sledeče dejavnike: • Seznam uporabniških zgodb je urejen in pripravljen za naslednjo iteracijo (cilj sestanka za izboljšavo seznama uporabniških zgodb). o Celotna razvojna ekipa mora opredeliti in oceniti velikosti uporabniških zgodb, da se izognemo dodatnim razpravam v sestanku za načrtovanje iteracije (Razvojna ekipa ne ocenjuje velikost uporabniških zgodb (SB24)). • Cilj naslednje iteracije je jasen (naloga lastnika izdelka, da pridobi potrebne želje od stranke in pripravi razumljiv cilj iteracije). • Skrbnik Scrum projekta poskrbi, da so pogoji za učinkovito izvedbo sestanka izpolnjeni (pripravljen dnevni red sestanka, rezervirana soba za sestanke, priprava pripomočkov, video konferenčna orodja so funkcionalna …).

	Ob tem je potrebno dnevni red sestanka sestaviti tako, da na najbolj učinkovit način predstavimo cilj iteracije, dodamo uporabniške zgodbe v seznam nalog iteracije in opredelimo podrobnosti, ki bodo pomagale razvojni ekipi razviti določeno uporabniško zgodbo. V primeru, ko sta priprava na sestanek in izvedba sestanka učinkovito izvedena, toda sestanek se še vedno podaljša, premislite, ali je vaša ekipa prevelika (Preseganje mej velikosti ekipe (SB7)). To lahko vpliva, da več članov ekipe deli svoje mnenje in se zaradi tega trajanje sestanka podaljša.

4.22 SB23: VODSTVO SPREMINJA SEZNAM NALOG ITERACIJE MED POTEKOM ITERACIJE

Tabela 28: Posledice in predlogi za izboljšavo SB23

Možne posledice	**Predlogi za izboljšavo**
Delo razvojne ekipe je zmoteno in cilj iteracije najverjetneje ne bo dosežen. To vpliva na povečanje nezadovoljstva v Scrum ekipi, saj ne uspejo doseči zastavljenega cilja. Izguba avtonomije razvojne ekipe, kar vpliva na manjšo motivacijo za delo.	Vodstvo mora razumeti, da s spreminjanjem seznama nalog iteracije zmoti delo razvojne ekipe, kar povzroči že prej omenjene posledice. Na določenih področjih dela lahko prihaja do bolj pogostih zahtev, ki morajo biti hitro upoštevane v Scrum projektu. Takšnemu načinu dela se lahko prilagodimo s krajšimi iteracijami. S krajšimi iteracijami lahko nove zahteve od stranke pravočasno dodamo v seznam nalog iteracije pri naslednjem sestanku za načrtovanje iteracije. Spreminjanje seznama nalog iteracije med iteracijo lahko stori le razvojna ekipa. Le-to stori v primeru, ko ugotovi, da mora biti seznam spremenjen za doseganje cilja iteracije. Razvojna ekipa lahko med iteracijo ugotovi, da izbrana tehnologija ne zadošča za doseganje cilja iteracije ali da so pozabili dodati nalogo v sestanku za načrtovanje iteracije, ki je ključna za doseganje cilja iteracije.

4.23 SB24: RAZVOJNA EKIPA NE OCENJUJE VELIKOST UPORABNIŠKIH ZGODB

Tabela 29: Posledice in predlogi za izboljšavo SB24

Možne posledice	**Predlogi za izboljšavo**
Sestanek za načrtovanje iteracije traja dlje, saj se razvojna ekipa ne strinja z oceno velikosti uporniških zgodb. Dodatne posledice o podaljševanju sestanka za načrtovanje iteracije lahko najdete pri ScrumBut-u: Podaljšanje trajanja sestanka za načrtovanje iteracije (SB22). Določeni člani razvojne ekipe, ki ne sodelujejo pri urejanju in ocenjevanju velikosti uporabniških zgodb, se ne zavedajo vsebine seznama uporabniških zgodb. To lahko vpliva, da del razvojne ekipe ne razume, katere naloge bodo razvite v prihodnje in zato primerno ne prilagodijo funkcionalnosti in arhitekturo sistema. V primeru, ko naloge ocenjujejo le strokovnjaki, obstaja velika verjetnost, da bodo naloge ocenjene preveč optimistično. Manj izkušeni razvijalci bodo najverjetneje porabili veliko več časa za razvoj v primerjavi z izkušenejšimi sodelavci.	Pri ocenjevanju nalog v seznamu uporabniških zgodb je pomembno, da so prisotni vsi člani razvojne ekipe. Vsi se morajo zavedati, kaj je vsebina seznama uporabniških zgodb in vsi morajo podati svoje mnenje glede ocene. V primeru, da so naloge ocenjene s strani drugih oseb, so lahko ocene napačne, saj obstaja možnost, da bodo druge osebe razvile to funkcionalnost. Različni razvijalci imajo različne izkušnje in spretnosti, zato je težko posploševati oceno le na peščici oseb. Ocenjevanje nalog iz seznama uporabniških zgodb je lahko zamudno dejanje, zato mora Scrum ekipa najti najbolj učinkovit način, kako se tega loti. Predlagamo, da si ekipe pogledajo metode, s katerimi povečajo učinkovitost ocenjevanja. Takšne metode so na primer Načrtovalni poker in T-shirt metoda. Pomembno je tudi, da ekipe ne ocenjujejo vse naloge v seznamu preveč podrobno. Podrobno ocenjevanje nalog, ki ne bodo kmalu razvite, ni smiselno, saj lahko razvoj drugih funkcionalnosti vpliva na oceno le-teh. Predlagamo, da bolj podrobno ocenite uporabniške zgodbe, ki bodo kmalu razvite, za ostale uporabniške zgodbe pa le podajte grobo oceno. Na ta način ne porabimo preveč časa na ponovnem ocenjevanju, ampak imamo približno oceno, kdaj bo možno določeno funkcionalnost ali projekt zaključiti in imamo dovolj informacij za razvrščanje nalog.

4.24 SB25: UPORABNIŠKE ZGODBE ZA SEZNAM NALOG ITERACIJE IZBERE VODSTVO

Tabela 30: Posledice in predlogi za izboljšavo SB25

Možne posledice	**Predlogi za izboljšavo**
Scrum ekipa ni zavezana, da bo dosegla cilj iteracije, saj sami niso izbrali nalog za seznam nalog iteracije. To vpliva na manjšo motivacijo članov ekipe in občutek, da sami nimajo vpliva pri odločitvah, kaj je smiselno narediti, da dosežejo najboljši rezultat. Pasivnost članov razvojne ekipe	Vodstvo v Scrum organizaciji se mora zavedati prej omenjenih posledic. Morajo sprejeti, da morajo za večjo uspešnost organizacije prilagoditi svoj način dela, ki so ga prej uporabljali v tradicionalni organizaciji. Scrum ogrodje temelji na samoorganiziranosti razvojnih ekip in vsak poseg v

vpliva na to, da Scrum proces ne bo dosegel svojega potenciala. Velikokrat vodstvo nima jasnega pregleda, kaj je potrebno narediti, da je zastavljen cilj dosežen, saj imajo premalo informacij, ki jih vedo le člani razvojne ekipe. Na primer, kako naloge v tehničnem dolgu (ang. technical debt) vplivajo na manjšo uspešnost razvoja naslednjih funkcionalnosti.	le-to povzroči negativen odziv razvojne ekipe. V primeru, da se vodstvo vpleta v naloge, ki so izključno namenjene razvojni ekipi, bodo razvijalci postali pasivni in Scrum proces ne bo dosegel svojega potenciala. Vodstvo nima dejanske vloge pri razvoju produkta v Scrum-u. Naloga vodstva pri Scrum-u je podpora: • lastnika izdelka z vpogledi in informacijami o viziji ter strategiji podjetja, • skrbnika Scrum projekta, da lahko spremeni organizacijske procese, ki bodo spodbujajo empiričnost, samoorganiziranost, transparentnost informacij in pametno izdajo produktov.

4.25 SB26: SEZNAM UPORABNIŠKIH ZGODB NI PREGLEDEN, NI UREJEN IN NEUSTREZNI ELEMENTI SEZNAMA UPORABNIŠKIH ZGODB NISO ODSTRANJENI.

Tabela 31: Posledice in predlogi za izboljšavo SB26

Možne posledice	**Predlogi za izboljšavo**
Razvojna ekipa nima jasnega pogleda na celoten produkt in izgubi osredotočenost, kaj prinese največjo vrednost za stranko. To lahko povzroči, da se najpomembnejše funkcionalnosti ne razvijejo najprej in da se razvijejo nepomembne funkcionalnosti. Enako posledico lahko vidimo pri ScrumBut-u: Imeti več lastnikov izdelka (SB1). Podaljševanje sestanka za načrtovanje iteracije ter sestanka za izboljšavo seznama uporabniških zgodb. Razvojna ekipa in lastnik izdelka se ne znajdejo v seznamu uporabniških zgodb ter ne morejo hitro ugotoviti, kaj je bolj pomembno za prihodnje iteracije in kaj manj. Enako posledico lahko vidimo pri ScrumBut-u: Imeti več lastnikov izdelka (SB1).	Za rešitev težav s tem ScrumBut-om predlagamo, da pogledate že prej omenjene predloge pri ScrumBut-u: Imeti več lastnikov izdelka (SB1).

4.26 SB29: SEZNAM UPORABNIŠKIH ZGODB NE OBSTAJA

Tabela 32: Posledice in predlogi za izboljšavo SB29

Možne posledice	**Predlogi za izboljšavo**
Obstaja velika verjetnost, da vizija produkta ni jasna in naloge, opredeljene s strani Scrum ekipe, ne odražajo dejanskih zahtev od stranke. Zaradi tega razvojna ekipa razvije produkt po njihovi intuiciji, kar lahko vodi do tega, da se razvijejo funkcionalnosti, ki imajo majhno vrednost za stranko in bi sicer bile na dnu seznama uporabniških zgodb ali popolnoma prezrte. To lahko pripelje do spreminjanja že izdelanih funkcionalnosti in veliko dodatnega dela, ko pridobijo povratne informacije od stranke. Le-to ustvari razočaranje, frustracijo, negativen vpliv na proračun, zamujene roke izdaje, slabšo kakovost produkta in manjšo dodano vrednost končnega produkta.	Predlagamo, da seznam uporabniških zgodb ustvarite, redno posodabljate in uporabljate kot oporo pri razvoju. Seznam uporabniških zgodb nam predstavi seznam zahtev, razvrščenih glede na poslovno vrednost. Le-to določi lastnik izdelka. V Seznam vsebuje vse elemente potrebne za uspešen razvoj produkta, vključno z hrošči, ki jih moramo odpraviti. Seznam uporabniških zgodb predstavlja povratne informacije iz različnih virov. Poleg tega vsebuje tudi naloge, ki so ključne samo za razvojno ekipo, katere stranka nikoli ne opazi pri končnem produktu. Seznam uporabniških zgodb nam olajša načrtovanje iteracije in izdajo produkta. V seznamu uporabniških zgodb so najbolj pomembni elementi vedno na vrhu seznama in tako razvojna ekipa ve, katere funkcionalnosti morajo dostaviti najprej. Pri delu s seznamom uporabniških zgodb moramo biti pozorni, da se nam ne zgodijo sledeči vzorci: • Lastnik izdelka uredi seznam uporabniških zgodb na začetku projekta, vendar ga ne posodablja, ko pridobiva nove informacije med potekom projekta. • Vse naloge v seznamu uporabniških zgodb so namenjene stranki. Seznam uporabniških zgodb ne vsebuje določenih nalog, potrebnih za doseganje uspešnega razvoja. • Seznam uporabniških zgodb je lokalno shranjen dokument in je redko deljen z ostalimi. To interesnim skupinam onemogoča, da bi prejemale posodobitve na seznamu.

4.27 SB30: RAZVOJ UPORABNIŠKIH ZGODB TRAJA DLJE OD TRAJANJA ITERACIJE

Tabela 33: Posledice in predlogi za izboljšavo SB30

Možne posledice	**Predlogi za izboljšavo**
Cilj iteracije ni pravočasno dosežen. Razvijalci imajo občutek, da lahko trajanje razvoja na uporabniški zgodbi podaljšujejo ter le-te dokončajo v naslednji iteraciji. Zaradi tega se ne naučijo, kako	V Scrum ekipah opazimo veliko različnih mnenj, ko pridemo do vprašanja, kaj storimo z nedokončano uporabniško zgodbo na koncu iteracije. Naš predlog je sledeč. Nedokončane uporabniške zgodbe na

boljše načrtovati razvoj uporabniških zgodb v prihodnih iteracijah, kar zna v prihodnjih iteracijah vplivati na manjšo zavzetost razvojne ekipe za doseganje cilja iteracije. Podaljševanje iteracije je pogosta posledica, če določene uporabniške zgodbe iz seznama nalog iteracije niso bile zaključene. Več podrobnostih v povezavi s podaljševanjem iteracij lahko vidite pri ScrumBut-u: Podaljševanje trajanje iteracije med iteracijo (SB21).	koncu iteracije vrnemo v seznam uporabniških zgodb in: • velikosti ocene uporabniške zgodbe ne spreminjamo, • uporabniške zgodbe ne razdelimo, da bi prikazali, da smo del uporabniške zgodbe končali v prejšnji iteraciji. Uporabniška zgodba je lahko ponovno dodana v naslednjo iteracijo, če je prioriteta le-te še vedno visoka (prioriteta te ali drugih uporabniških zgodb se lahko spremeni). Razvojna ekipa se mora čimbolj osredotočiti na dostavljanje novih funkcionalnosti ter s tem največje vrednosti za stranko brez dodatnega nepotrebnega dela. Sama razdelitev ali spreminjanje velikosti uporabniške zgodbe prinese dodatno delo in nobene koristi pri končnem produktu, zato se moramo temu izogibati. Naloga razvojne ekipe je določiti smiselno velikost uporabniške zgodbe pred začetkom iteracije. V primeru, da je to storila pravilno, je razdelitev nesmiselna, saj manjši deli funkcionalnosti ne bodo prinesli nove uporabne funkcionalnosti. Vodstvo ali stranka so pogosto osredotočeni na dosežene točke na koncu iteracije. Njim moramo obrazložiti, da je vaš način dela umerjen k doprinosu največje dolgoročne funkcionalnosti na produktu in ne toliko k podrobnemu poročanju, saj na ta način ne zagotovite najučinkovitejšega načina razvoja produkta.

4.28 SB31: DEFINICIJA KONČANOSTI NE OBSTAJA

Tabela 34: Posledice in predlogi za izboljšavo SB31

Možne posledice	**Predlogi za izboljšavo**
Standard za kakovost opravljenega dela v uporabniških zgodbah ni opredeljen. Le-to povzroča, da določeni koraki za preverjanje kakovosti niso narejeni na koncu razvoja uporabniške zgodbe. Zaradi tega lahko nastane nedoslednost razvoja uporabniških zgodb pri različnih razvijalcih in dolgoročno več težav s kakovostjo izdelka. Poveča se število predelav in stranki ne zagotovimo popolnoma delujoč inkrement produkta na koncu iteracije.	Definicija končanosti je eden od ključnih elementov v Scrum procesu za zagotavljanje primerne kakovosti in doslednosti pri razvoju uporabniških zgodb. Priporočamo, da jo opredelite na nivoju podjetja ali vsaj na nivoju Scrum ekipe. Definicija končanosti predstavlja formalno opredelitev kakovosti, ki jo mora doseči vsaka razvita uporabniška zgodba. Na primer pri razvoju programske opreme lahko definicija končanosti opredeli, da mora vsaka uporabniška zgodba biti: • razvita po opredeljenem kodirnem standardu, • pregledana s strani drugega razvijalca,

	• preizkušena s strani preizkuševalca, • preizkušena z avtomatskimi testi, • dokumentirana. Definicija končanosti zagotavlja, da vsi člani v ekipi natančno vedo, kaj je pričakovano od vsake uporabniške zgodbe, ki jo razvojna ekipa dostavi stranki. Na ta način zagotovimo stabilen inkrement na koncu iteracije in manj pritožb s strani uporabnikov. V primeru, ko uporabniška zgodba ne izpolni vseh pogojev definicije končanosti na koncu iteracije, se ta uporabniška zgodba dojema kot nedokončana in je ne predstavimo na sestanku za predstavitev rezultatov iteracije. V okolju, kjer več Scrum ekip sodeluje pri razvoju izdelka, morajo vzajemno opredeliti definicijo končanosti in jo skupaj upoštevati.

4.29 SB32: NA KONCU ITERACIJE NI POVEČANJA FUNKCIONALNOSTI NA PRODUKTU

Tabela 35: Posledice in predlogi za izboljšavo SB32

Možne posledice	**Predlogi za izboljšavo**
Testiranje se v določenih ekipah opravi v naslednji iteraciji, kar povzroči, da se razvita funkcionalnost v trenutni iteraciji ne doda takoj h končnemu produktu. Pri tem vzorcu nastanejo dodatne posledice, kot so: • nova koda je napisana na podlagi stare nepreizkušene kode, kar lahko vodi k večjemu številu popravkov na novi kodi; • kasnejši rezultati testiranja vplivajo na to, da bo zahteva za odpravo hroščev prišla veliko kasneje od končanega razvoja na določeni funkcionalnosti. Le-to povzroča, da razvijalec porabi več časa, da se spomni kako je bila funkcionalnost razvita in preizkuševalec porabi več časa za obrazložitev težav razvijalcu. Takšen pristop za odpravo hroščev ni najučinkovitejši; • večji popravki ali večje število popravkov na funkcionalnostih iz prejšnje iteracije lahko zmotijo delo razvijalcev do te mere, da ne morejo doseči cilja trenutne iteracije.	Glede na prej omenjene posledice bi vam priporočali, da ne uporabljate: • ločenega testiranja funkcionalnosti iz prejšnje iteracije, saj zmanjša učinkovitost dela razvojne ekipe in podaljša čas za pridobitev povratnih informacij; • dostavljanja funkcionalnosti stranki takoj, ko so razvite, saj lahko prinese dodatne težave s strankino prilagoditvijo na interval posodobitev in dodatne težave pri komunikaciji s stranko; • uporabe posebnih iteracij, saj nasprotujejo temeljnemu namenu iteracije. Le-ta je ustvarjanje novega inkrementa produkta, hitro pridobitev povratnih informacij in hitra prilagoditev. Povečanje funkcionalnosti na produktu na koncu vsake iteracije je osnovano na načelu, da se čimbolj poveča uporaba metode pregleda in prilagajanja. Na ta način vse interesne skupine vedo, kaj lahko pričakujejo na koncu iteracije, pregledajo napredek in se dogovorijo, kako bo potekal razvoj v prihodnje.

V primeru, ko se novo razvite funkcionalnosti dodajajo med iteracijo, je lahko interval posodobitev prehiter za stranko in uporabnike. Stranka se lahko težko sooča z dejstvom, da funkcionalnosti ne pregleda in potrdi, preden so dodane v produkcijo. Tudi samo izobraževanje uporabnikov je težje izvedljivo, saj nove funkcionalnosti niso dostavljene le ob določenih datumih (npr. na koncu vsake iteracije). Uporaba posebnih iteracij, kjer cilj ni povečanje funkcionalnosti na produktu, zmanjša razumevanje samega smisla iteracije in nasprotuje načelom Scrum-a. Iteracija je najboljši način, da s povečanjem funkcionalnosti predstavimo inkrement produkta, ga pregledamo in se glede na videno prilagodimo. Določene iteracije, ki niso smiselno uporabljene, so sledeče: • Iteracija 0 temelji na ustvarjanju vizije in začetnega seznama uporabniških zgodb. Leto je pomembno na začetku razvoja, ampak se ne bi smelo imenovati iteracija, saj ne prinese povečanja funkcionalnosti na produktu, kar je temeljni cilj vsake iteracije. • Oblikovalna iteracija (ang. design sprint) je namenjena ustvarjanju tehnične arhitekture, ki bo upoštevana v prihodnjih iteracijah. Takšna iteracija ne le, da ne prinese povečanja funkcionalnosti na produktu, ampak tudi zmanjša metodo pregleda in prilagajanja v prihodnjih iteracijah, saj je arhitektura že zadana. Takšen način dela je tipičen pri slapovni metodi v tradicionalnih organizacijah. Enostavna tehnična arhitektura se lahko ustvari med potekom tipične iteracije in se posodablja v prihodnjih iteracijah, saj lahko zmanjša tveganja in poenoti razvijalce pri razvoju. Potrebno pa se je izogibati ustvarjanja bolj podrobnega in končnega načrta arhitekture, kar je namen oblikovalne iteracije. • Utrjevalna iteracija (ang. hardening sprint) je osredotočena za dokončanje funkcionalnosti iz prejšnje iteracije, ki niso dosegle definicije končanosti. Uporaba takšne iteracije nakazuje na nepopolno	Cilj je doseči majhne, ampak konstantne korake, s katerimi poskrbimo za največjo uporabno vrednost za stranko. Zato je zelo pomembno, da se vsaka Scrum ekipa drži načela, da se v vsaki iteraciji poveča funkcionalnost produkta, ki bo prinesel dodano vrednost za stranko.

uporabo Scrum ogrodja, saj se razvojna ekipa sooča s težavo doseganja cilja iteracije. Tak način dela zmanjša uporabnost metode pregleda in prilagoditve, saj vseh predvidenih funkcionalnosti ne dostavimo pravočasno. Uporaba takšne iteracije nakazuje, da Scrum ekipa nima vsega potrebnega, da ustvari potencialno dostavljiv produkt.	

4.30 SB33: EKIPE NE UPORABLJAJO PADAJOČI DIAGRAM SPREMLJANJA NAPREDKA

Tabela 36: Posledice in predlogi za izboljšavo SB33

Možne posledice	**Predlogi za izboljšavo**
Scrum ekipa potrebuje več časa, da opazi določene težave pri razvoju. Zaradi tega se ekipa kasneje prilagodi in ne doseže optimalne učinkovitosti. To lahko vodi v podaljševanje rokov, manj dostavljenih funkcionalnosti in slabši kakovosti končnega produkta. Dodatne posledice so opisane pri ScrumBut-u: Padajoči diagram spremljanja napredka ni viden celotni ekipi (SB34).	Pri uporabi Scrum-a je priporočeno uporabljati padajoči diagram spremljanja napredka, saj prej odkrijemo težave z različnimi vzorci na diagramu. Različne vzorce diagrama in njihov pomen lahko najdete podrobneje opisano v poglavju 2.2.1.

4.31 SB34: PADAJOČI DIAGRAM SPREMLJANJA NAPREDKA NI VIDEN CELOTNI EKIPI

Tabela 37: Posledice in predlogi za izboljšavo SB34

Možne posledice	**Predlogi za izboljšavo**
Nepreglednost napredka s padajočim diagramom spremljanja napredka posledično zmanjša ozaveščenost razvojne ekipe o trenutnem stanju opravljenih in neopravljenih uporabniških zgodb v iteraciji. Zaradi tega lahko razvojna ekipa misli, da je njihov napredek dober in ne opazijo težav drugih sodelavcev, katerim bi drugače lahko ponudili pomoč. Razvojna ekipa nima pred seboj padajočega diagrama spremljanja napredka in zato ne ve,	Za doseganje največje preglednosti napredka predlagamo, da padajoči diagram spremljanja napredka natisnete na začetku iteracije, ga dodate na tablo z nalogami. Le-tega ročno posodabljate pri vsaki spremembi v iteraciji. Na ta način poskrbimo, da ima Scrum ekipa jasno preglednost nad potekom iteracije. Preostale interesne skupine lahko hitro obvestimo o napredku poteka iteracije s sliko diagrama. Uporaba digitalnega diagrama je tudi ena od možnosti, kako doseči večjo preglednost med vsemi interesnimi skupinami. Ne glede katero obliko

koliko dela lahko opravijo v eni iteraciji. Zaradi tega se poveča verjetnost, da razvojna ekipa ne bo pravilno načrtovala količine dela v prihodnjih iteracijah. Interesne skupine se ne zavedajo, da ima razvojna ekipa težave z razvojem. Skrbnik Scrum projekta ne opazi, da določena uporabniška zgodba povzroča dodatne težave razvojni ekipi.	uporabimo, moramo poskrbeti, da je diagram redno posodobljen.

4.32 SB35: RAZVOJNA EKIPA NI SAMOORGANIZIRANA

Tabela 38: Posledice in predlogi za izboljšavo SB35

Možne posledice	**Predlogi za izboljšavo**
Razvojna ekipa ni samo-organizirana, saj ima težave s komunikacijo. Le-to vpliva na: • povečane stroške, saj bo razvojna ekipa porabila več ur za opredelitev obsega dela in zagotovitev, da so vse zahteve dosežene, • več zamud na projektih, saj je potrebno več koordinacije med člani razvojne ekipe, • slabšo kakovost produkta zaradi slabše izmenjave znanj in manj kakovostne komunikacije med člani razvojne ekipe. Razvojna ekipa ni samoorganizirana, saj ima težave s sodelovanjem. Le-to vpliva na: • slabšo kakovost produkta, saj ekipa ni sposobna razviti odnos, ki bi bil vsestransko prilagodljiv; • slabšo vrednost za stranko, saj obstaja večja nevarnost, da člani razvojne ekipe ne bodo medsebojno delili informacije in pravilno dojeli zahteve od stranke; • večje stroške, saj informacije niso deljene na najučinkovitejši način, kar vodi v slabše razumevanje zahtev in večje število ur, potrebnih za razvoj; • večje stroške za komponente strojne opreme, saj zaradi slabšega sodelovanja ekipa pozno zaključi načrt, ki lahko vpliva na pospešeno (dražjo) dobavo komponent z zunanjim dobaviteljem;	Samoorganizirane ekipe se same najbolje odločijo ,kako opraviti svoje delo. Veliko bolje kot da bi jih usmerjali drugi člani izven ekipe. Večja verjetnost je, da bo ekipa bolj samoorganizirana, če: • so vloge jasno in preprosto opredeljene, • se člani v Scrum ekipi in v podjetju razumejo in spoštujejo, • so vsi člani odgovorni za svoje naloge v vlogi, • vsi člani ekipe razumejo Scrum vrednote in ustvarijo zaupanja vredno okolje, idealno za hitro učenje, • se retrospektivni sestanki izvajajo zelo kakovostno z usmerjenostjo za izboljšanje načina dela. Obstaja veliko nasvetov, kako zgraditi in spodbujati samoorganizirane ekipe. Le-ti vključujejo ugotavljanje skupnega namena, odstranjevanje upravljajočega načina vodenja ter ustvarjanje varnega okolja. Kljub temu včasih člani ekipe ne morejo pozabiti na svoje zakoreninjene običaje, ki jim otežujejo, da bi popolnoma sprejeli samoorganizacijski način dela. Ljudje imamo različno kulturno ozadje, navade, prepričanja, interese, sposobnosti, znanja, spretnosti in temperamente. Bolj izkušeni člani v ekipah pogosto preglasijo mlajše sodelavce, kar vpliva na slabšo samoorganizacijo. Mlajši sodelavci pogosto postavljajo pravilna vprašanja, vendar jih velikokrat ne znajo dobro ubesediti in nehote popustijo, ko

• Zamik izdaje produkta, saj se informacije ne bodo delile na najučinkovitejši način, kar povzroča zamude pri dobavi posameznih komponent. Razvojna ekipa ni samo-organizirana, saj ima težave s cenjenjem ljudi. Le-to vpliva na slabšo izvedbo projekta zaradi slabšega sodelovanja, komunikacije, kakovosti in stroškov.	starejši člani odločno podajo svoje argumente. Pomembno je, da so vsi člani v ekipi na istem nivoju, brez oznak, kjer nobena hierarhija ni spodbujena. Na ta način se lahko ekipa združi kot enota in odkrito izrazi svoja stališča in sodeluje pri odločanju. Ekipe bi morale biti še posebej potrpežljive in empatične z mlajšimi člani, saj jim na ta način pomagajo, da čim več prispevajo h končnemu rezultatu. Ko v ekipi prepoznamo ovire, moramo člane spodbuditi, da se soočijo s težavami in s tem še izboljšajo delovni odnos. Sodelovati morajo kot celotna ekipa, če želijo doseči najboljši rezultat. Najboljši rezultat dosežejo ekipe, ki v primeru nedosegljivosti lastnika izdelka in skrbnika Scrum projekta same prevzamejo odgovornost in s kritičnim razmišljanjem najdejo rešitev, kako nadaljevati z razvojem.

4.33 SB36: PRETIRANA UPORABA METRIK (NPR. SPREMLJANJE KLJUČNIH KAZALCEV UČINKOVITOSTI – KPI)

Tabela 39: Posledice in predlogi za izboljšavo SB36

Možne posledice	**Predlogi za izboljšavo**
Scrum ekipa se osredotoča na delo, ki bo prikazalo dobre kazalce in ne na delo, ki bo prineslo največjo dodano vrednost za stranko. Pritisk vodstva zaradi nepravilno uporabljenih metrik poveča pritisk na Scrum ekipo, kar zmanjša kakovost produkta in povzroča izgorelost članov ekipe.	Pravilna uporaba metrik je lahko zelo koristna pri razvoju produkta. Dobra metrika je tista, ki nam pove nekaj koristnega, ne da povzroči škodo. Pomembno se je zavedati, da lahko določene metrike uporabimo na napačen način in s tem preusmerimo osredotočenost ekipe. Na primer določene ekipe bodo preusmerile osredotočenost razvoja tako, da bo prikaz matrik dober. Le-to ne pomeni, da prinese največjo dodano vrednost za stranko. Za primer vzemimo metriko za količino opravljenega dela v iteraciji. Za razvojno ekipo je to pomembna metrika, saj z njo preverijo ali izboljšujejo svojo uspešnost v primerjavi s prejšnjimi iteracijami. Je tudi zelo uporabna vrednost za načrtovanje različnih izdaj produkta. Po drugi strani je to zelo slaba metrika, če ga uporabi vodstvo za merjenje in vrednotenje uspešnosti ekipe ali posameznika. Še slabše je, če vodstvo uporabi to metriko za primerjanje uspešnosti različnih ekip, kjer vsaka Scrum ekipa količino opravljenega dela opredeli po svojih pogojih. Na primer zaradi metrik lahko opazimo vzorec, ko vodstvo začne pritiskati na Scrum ekipo, da pospeši delo v prihodnjih iteracijah. To lahko pripelje do

	sledečih nasprotujočih si rezultatov: • ekipa začne delati več, kar zmanjša kakovost produkta ali povzroči, da člani ekipe izgorijo; • ekipa spremeni definicijo končanosti, kar omogoči več opravljenega dela v iteraciji, toda slabšo kakovost izdelka; • ekipa spremeni referenčno oceno za uporabniške zgodbe, kar pomeni, da opravi enako količino dela, toda le-to prikažejo z večjo vrednostjo; • ekipa čuti več pritiska, kar zmanjša motivacijo razvojne ekipe in s tem tudi količino opravljenega dela v iteraciji. Pomembno je, da podjetja izvajajo meritve kazalnikov učinkovitosti na neinvaziven način, razvojnim ekipam pa naj jasno razložijo namen merjenja, saj lahko drugače ekipa meni, da je namen merjenja kaznovanje, če uspešnost ni dosežena.

4.34 SB37: UPORABNIŠKE ZGODBE SO OPREDELJENE HORIZONTALNO IN NE VERTIKALNO

Tabela 40: Posledice in predlogi za izboljšavo SB37

Možne posledice	**Predlogi za izboljšavo**
Ni možno pridobiti povratnih informacij po vsaki razviti uporabniški zgodbi. Horizontalno opredeljene uporabniške zgodbe morajo biti združene, da lahko stranka preveri funkcionalnost in poda Scrum ekipi povratne informacije. To lahko vpliva na daljši cikel pridobivanja povratnih informacij in s tem več možnosti za kasnejše predelave funkcionalnosti. Težave s komunikacijo med razvijalci lahko povzročijo, da združene horizontalne uporabniške zgodbe potrebujejo dodatne dodelave, da zadovoljimo potrebe stranke. Različni razvijalci so po navadi zadolženi za razvoj posameznih horizontalno opredeljenih uporabniških zgodb, ki skupaj ustvarijo funkcionalnost za stranko. Takšen način dela zahteva veliko povezovanja med člani ekipe, kar predstavlja veliko možnosti za napake. V primeru, ko sporazumevanje med člani ekipe ni najboljše, lahko horizontalno opredeljene uporabniške zgodbe vodijo tudi do nedoseženega	Pri uporabi Scrum-a je zelo priporočljivo deliti funkcionalnosti vertikalno in ne horizontalno. Horizontalna opredelitev pri razvoju programske opreme pomeni, da razvijemo podatkovno bazo, vmesnike in grafični vmesnik v treh posameznih uporabniških zgodbah. Vertikalna opredelitev nam omogoči, da v vsaki uporabniški zgodbi razvijemo vse arhitekturne nivoje (podatkovno bazo, vmesnike in grafični vmesnik), v manj obsežni, toda delujoči funkcionalnosti. Na takšen način vsaka uporabniška zgodba prinese dodano vrednost za stranko, saj so razviti vsi potrebni deli, da je funkcionalnost lahko uporabljena. Poleg tega stranka lahko funkcionalnost pregleda in nam hitro poda povratne informacije. Dolgoročno se na takšen način lahko izognete dodatnim zamudam na projektu. Pri delitvi uporabniških zgodb moramo upoštevati sledeče dejavnike: • uporabniške zgodbe morajo biti dovolj majhne, da so lahko končane v eni iteraciji;

cilja iteracije.	• vsaka uporabniška zgodba mora prinesti dodano vrednost za stranko. Zaradi tega je INVEST v Scrum-u pogosto uporabljena metoda, ki opredeli, kako mora biti uporabniška zgodba opredeljena: • neodvisna (ang. independent) • spremenljiva po dogovoru (ang. negotiable), • koristna (ang. valuable), • ocenljiva (ang. estimable), • majhna (ang. small), • preverljiva (ang. testable).

5 VPRAŠALNIK – INTEGRIRAN V SPLETNO STRAN

5.1 NAMEN SPLETNE STRANI

Kot že omenjeno v uvodu, je končni cilj magistrske naloge ustvariti spletno stran, na kateri uporabniki rešijo spletni vprašalnik in na podlagi odgovorov pridobijo informacije, kaj je smiselno spremeniti pri njihovi uporabi Scrum-a.

Spletna stran je dostopna na povezavi: https://student.famnit.upr.si/~89142035/

5.2 SESTAVA SPLETNE STRANI

Spletna stran je sestavljena iz petih strani:

- domov,
- spletni vprašalnik,
- razlaga ScrumBut-ov,
- o meni,
- politika zasebnosti.

5.2.1 **Stran »Domov«** (https://student.famnit.upr.si/~89142035/):

Uvodna stran, ki se prikaže, ko uporabnik odpre spletno stran. Stran je namenjena pregledu vsebine celotne spletne strani.

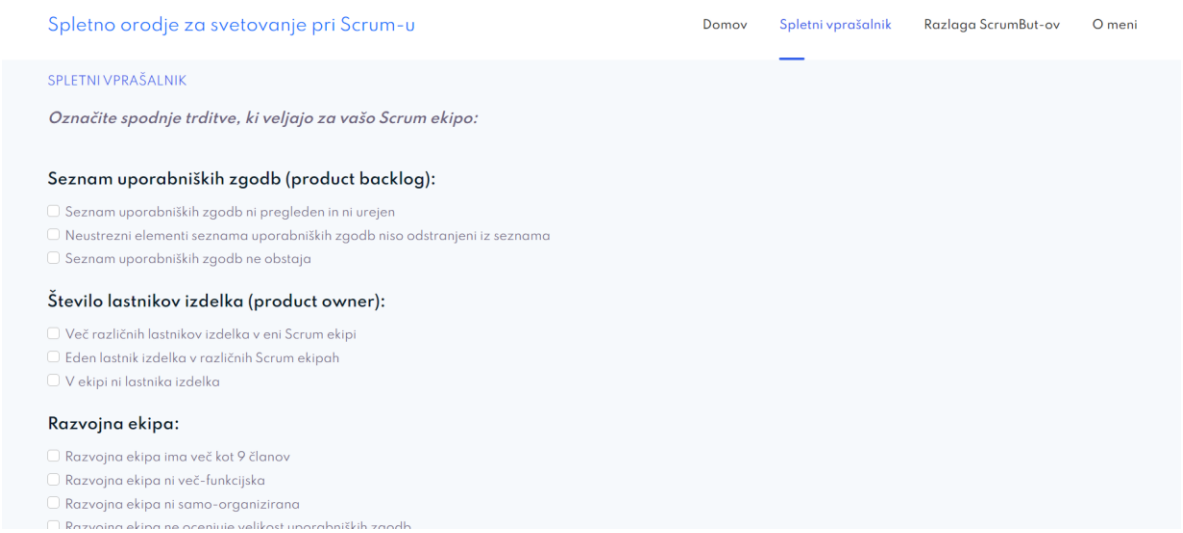
Slika 11: Stran »Domov«

5.2.2 **Stran »Spletni vprašalnik«** (https://student.famnit.upr.si/~89142035/spletni-vprasalnik/):

Stran, kjer uporabniki v seznamu označijo vsa težavna območja, s katerimi se soočajo pri uporabi Scrum-a. Ko to storijo, lahko pritisnejo na gumb »Prikaži prepoznane ScrumBute«, ki jih preusmeri na stran »Razlaga ScrumBut-ov«. Tam uporabniki pridobijo teoretično podlago in predloge za izboljšavo v povezavi s prepoznanimi ScrumBut-i.

Spletni vprašalnik smo razdelili na sledeče kategorije, da uporabniki lažje najdejo področja, s katerimi imajo največ težav:

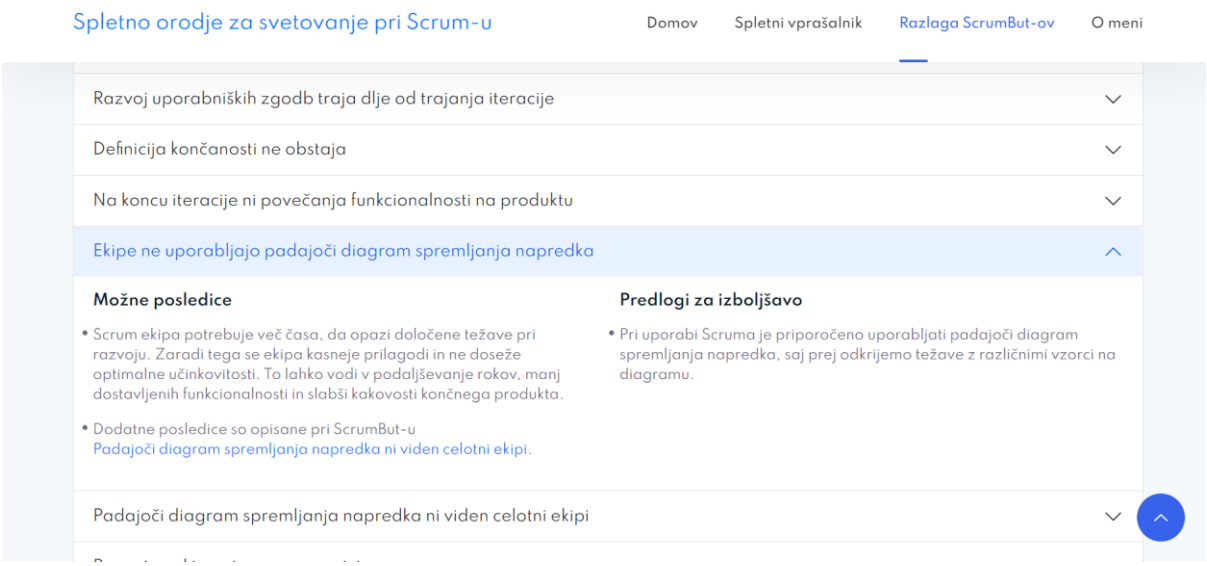
- seznam uporabniških zgodb (ang. product backlog),
- število lastnikov izdelka (ang. product owner),
- razvojna ekipa,
- dnevni sestanki,
- preostali Scrum sestanki,
- iteracije,
- nepravilna struktura Scrum ekipe,
- vpletanje vodstva v delovanje Scrum ekipe,
- padajoči diagram spremljanja napredka (ang. burn-down chart),
- ostala področja.



Slika 12: Stran »Spletni vprašalnik«

5.2.3 **Stran »Razlaga ScrumBut-ov«** (https://student.famnit.upr.si/~89142035/razlaga-scrumbutov/):

Na strani so za vsak posamezen ScrumBut predstavljene posledice ScrumBut-a in konkretni predlogi za izboljšavo, ki lahko uporabnikom pomagajo, da težave s ScrumBut-om hitreje rešijo. V primeru, da uporabnik na stran pride s strani »Spletni vprašalnik«, se bodo uporabniku prikazali le ScrumBut-i, ki so bili prepoznani iz odgovorov na prejšnji strani. V tem primeru uporabnik še vedno lahko prikaže razlago vseh ScrumBut-ov s pritiskom na gumb »Prikaži vse ScrumBut-e«.



Slika 13: Stran »Razlaga ScrumBut-ov«

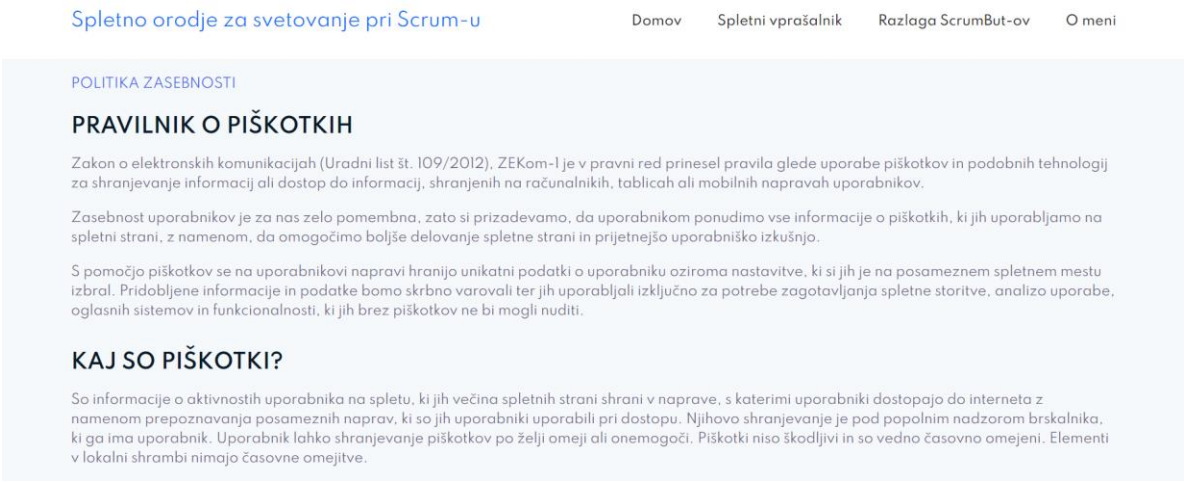
5.2.4 **Stran »O meni«** (https://student.famnit.upr.si/~89142035/o-meni/):

Namen strani je predstavitev podatkov o avtorju.

Slika 14: Stran »O meni«

5.2.5 **Stran »Politika zasebnosti«** (https://student.famnit.upr.si/~89142035/politika-zasebnosti/):

Zakon o elektronskih komunikacijah (Uradni list št. 109/2012), ZEKom-1, zahteva, da na spletni strani uporabnikom predstavimo, zakaj se piškotki na spletni strani uporabljajo. Na strani »Politika zasebnosti« smo zato prikazali vse potrebne podatke, da uporabnik razume, zakaj se piškotki uporabljajo in koliko časa bodo hranjeni.



Slika 15: Stran »Politika zasebnosti«

5.3 OPIS TEHNOLOGIJE ZA IZDELAVO SPLETNE STRANI

Spletna stran je razvita po principu progresivne spletne aplikacije z uporabo tehnologij html5, css3/scss in javascript. Takšen princip nam omogoča boljšo kompatibilnost z različnimi brskalniki in mobilnimi napravami. Obnašanje aplikacije je tudi boljše v primeru slabe internetne povezave, saj aplikacija že ogledano vsebino hrani v predpomnilniku. Za poenostavitev izdelave smo uporabili Bootstrap 5 ogrodje za izdelavo odzivne spletne strani in predlogo Masive (https://graygrids.com/templates/massive-multipurpose-html-bootstrap-template/), ki je osnovana na Bootstrap 5 ogrodju.

6 ZAKLJUČEK

Vse pogosteje se podjetja odločajo za uporabo Scrum-a z namenom povečanja poslovne učinkovitosti. Načela Scrum ogrodja so enostavna, toda težko izvedljiva zaradi veliko različnih vplivov. Z največjim številom težav se soočajo podjetja, ki vpeljujejo Scrum. Soočajo se s spremembo kulture, strategije in strukture podjetja ter s spremembo vlog, delovnih postopkov, orodij in komunikacije znotraj ekipe ter s stranko. Scrum vsebuje veliko različnih metod, sestankov in določenih pravil. Določene osebe v podjetjih menijo, da niso ustrezna za njih in jih prilagodijo brez razumevanja posledic. Naše mnenje je, da je to ena od največjih težav za doseganje učinkovite uporabe Scrum-a.

Ker se zavedamo, da je doseganje učinkovite uporabe Scrum-a zelo težko doseči, smo v spletni strani predstavili posledice in predloge za izboljšavo najbolj težavnih področij. Na ta način želimo podjetjem pomagati, da hitro pridejo do bistvenih informacij in s tem do želene poslovne učinkovitosti z uporabo Scrum-a.

7 VIRI IN LITERATURA

[1] L. Adkins, *Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition*. Addison-Wesley, 2010.

[2] P. S. Adler in A. J. Shenhar, "Adapting your technological base: The organizational challenge," *Sloan Management Review*, vol. 32, št. 1, str. 25–37, 1990.

[3] G. Asproni, "An Introduction to Scrum". *Software Developer's Journal*, str. 1–10, 2006.

[4] K. Beck, *Extreme Programming Explained: Embrace change*, Upper Saddle River, New Jersey: Addison-Wesley, 2000.

[5] B. Boehm in R. Turner, *Balancing Agility and Discipline: A Guide for the Perplexed*, Boston: Addison-Wesley, 2004.

[6] B. Boehm in R. Turner, "Management Challenges to Implementing Agile Processes in Traditional Development Organizations", *IEEE Software*, vol. 22, str. 30–39, 2005.

[7] R. B. Bostrom in B. D. Thomas, "Achieving excellence in communications: A key to developing complete, accurate and shared information requirements", v *The Proceedings of the Twentieth Annual Computer Personnel on Research Conference*. Charlottesville, VA, USA, 1983, 1–13.

[8] L. D. Brown, M. E. Feinberg in M. L. Kan, "Predicting engagement in a transition to parenthood program for couples", *Evaluation and Program Planning*, vol. 35, št. 1, str. 1–8, 2012.

[9] V. Butkus, "Too Big to Scale – Optimal Scrum Team Size Guide", toptal. Dosegljivo: https://www.toptal.com/product-managers/agile/scrum-team-size.

[10] S. Cavaleri in K. Obloj, *Management Systems: A Global Perspective*, Belmont, CA, USA: Wadsworth Publishing Company, 1993.

[11] A. Cockburn, *Agile Software Development*, USA, Indianapolis: Addison-Wesley, 2002.

[12] M. Cohn in D. Ford, "Introducing an Agile Process to an Organization", *Computer*, vol. 36, str. 74–78, 2003.

[13] M. Cohn. (2009). "Succeeding with Agile: Software Development Using Scrum". Dosegljivo: https://www.researchgate.net/publication/234803335_Succeeding_with_Agile_Software_Development_Using_Scrum.

[14] S. Counsell et al., "What Formal Models Cannot Show Us: People Issues During The Prototyping Process", v *The proceedings of the 6th International Conference on Product Focused Software Process Improvement.* Oulu, Finland, 2005, 3–5.

[15] E. Derby in D. Larsen, *Agile retrospectives: Making good teams great*. Raleigh, NC: Pragmatic Bookshelf, 2006.

[16] A. Drægert in D. Petersen, "ScrumBut in Professional Software Development", Magistrska naloga. Aalborg, Aalborg university, Department of Comuter Science, 2016.

[17] S. Denning (4. 11. 2018). *The 12 Stages Of The Agile Transformation Journey*. Dostopno: https://www.forbes.com/sites/stevedenning/2018/11/04/the-twelve-stages-of-the-agile-transformation-journey/#640256083dd4.

[18] A. Edstrom, "User Influence and the Success of MIS Projects: A Contingency Approach", *Human Relations*, vol. 30, št. 7, str. 580–607, 1997.

[19] V. Eloranta, K. Koskimies in T. Mikkonen, "Exploring ScrumBut – An Empirical Study of Scrum Anti-Patterns", *Information and Software Technology*, vol. 74, str. 194–203, 2015.

[20] M. Fowler (18. 7. 2006). *Using an Agile Software Process with Offshore Development*. Dostopno: http://www.martinfowler.com/articles/agileOffshore.html.

[21] T. J. Gandomani et al., "Towards Comprehensive and disciplined Change Management Strategy in Agile Transformation Process", *Research Journal of Applied Sciences, Engineering and Technology*, vol. 6, str. 2345–2351, 2012.

[22] T. J. Gandomani et al., "Exploring Facilitators of Transition and Adoption to Agile Methods: A Grounded Theory Study", *Journal of software*, vol. 9, št. 7, str. 1666–1678, 2014.

[23] T. P. Gustavo (2016). "Scrum vs ScrumAnd vs ScrumBut: which one are you doing?" *Agile Portugal 2016*. Dosegljivo: https://www.slideshare.net/pedrogustavotorres/scrum-vs-scrumand-vs-scrumbut-which-one-are-youdoing-agile-portugal-2016

[24] J. R. Hackman, M. S. Pallack in R. O. Perloff, "The psychology of self-management in organizations", *Psychology and work: Productivity, change, and employment*, str. 89–136, 1986.

[25] G. Hamel, "Moon shots for management (reinventing management to make it more relevant to a volatile business environment)", *Harvard Business Review*, vol. 87, št. 2, str. 91–99, 2009.

[26] S. Hanly, L. H. Waite, L. Meadows, in R. Leaton. "Agile coaching in British Telecom: making strawberry jam". *AGILE 2006 Conference*, Minneapolis, MN, USA, 2006, str. 194–202.

[27] O. Hazzan, in Y. Dubinsky. "Coaching agile software projects: tutorial proposal*". 32nd International Conference on Software Engineering (ICSE), Cape Town, 2010, str. 481–482.*

[28] J. Highsmith, (2003). *Cutter Consortium Reports: Agile Project Management Principles and Tools*. Dostopno: https://www.cutter.com/article/agile-project-management-principles-and-tools-424511.

[29] J. Highsmith in A. Cockburn, "Agile Software Development: The Business of Innovation", *IEEE Computer*, vol. 30, str. 120–122, 2001.

[30] K. H. Judy, I. Krummins-Beens, "Great Scrums Need Great Product Owners: Unbounded Collaboration and Collective Product Ownership". *41st Hawaii International Conference on System Sciences, 2008.*

[31] M. Korkala, P. Abrahamsson in P. A. Kyllönen, "Case Study on the Impact of Customer Communication on Defects in Agile Software Development". *AGILE 2006 Conference, Minneapolis, MN, USA, 2006, str. 76–88.*

[32] M. Korkala, M. Pikkarainen in K. Conboy, *Agility Across Time and Space: Combining Agile and Traditional: Customer Communication in Distributed Environment*, str. 201–216, V D. Šmite, N. B. Moe in P. J. Åkerfalk (urednik). Germany: Springer, 2010.

[33] M. Korkala, "Customer communication in distributed agile software development", Doktorska disertacija. Espoo, University of Oulu, 2015.

[34] J. P. Kotter, "Seizing opportunities and dodging threats with a dual operating system", *Strategy & Leadership*, vol. 42, št. 6, str. 10–12, 2014.

[35] I. Krasteva in S. Ilieva, "Adopting an agile methodology: why it did not work", *Proceedings of the 2008 international workshop on Scrutinizing agile practices or shoot-out at the agile corral*, str. 33–36, 2008.

[36] V. Krishna, in A. Basu, "Scrum+:: Is it "ScrumBut" or "ScrumAnd"". 2011 *Annual IEEE India Conference, 2011*, str. 1–4.

[37] G. J. Miller, (2013). "Agile problems, challenges and failures", v *PMI® Global Congress*. New Orleans: Project Management Institute, 2013.

[38] S. Nerur, R. Mahapatra, in G. Mangalaraj, "Challenges of migrating to agile methodologies", *Communications of the ACM*, vol. 48, št. 5, str. 72–78, 2005.

[39] J. Noll et al., *A Study of the Scrum Master's Role*, 2017.

[40] M. Saeki, "Communication, Collaboration and Cooperation in Software Development – How Should We Support Group Work In Software Development? ", *the proceedings of the 2nd Asia-Pacific Software Engineering Conference*, str. 12–20, 1995, Brisbane, Queensland, Australia.

[41] K. Schwaber, in M. Beedle, *Agile Software Development with SCRUM*. New York: Prentice Hall, 2001.

[42] K. Schwaber, in J. Sutherland (2020). *The 2020 Scrum Guide*. Dostopno: https://www.scrumguides.org/scrum-guide.html

[43] Y. Shastri, R. Hoda in R. Amor, "Does the "Project Manager" Still Exist in Agile Software Development Projects?", *23rd Asia-Pacific Software Engineering Conference (ASPEC)*, str. 57–64. 2016.

[44] A. Sillitti, M. Ceschi, B. Russo, in G. Succi, "Managing uncertainty in requirements: a survey in documentation-driven and agile companies", *11th IEEE International Symposium on Software Metrics*, str. 10–17. Como, Italy, 2005.

[45] V. Stray, N. B. Moe in D.I.K. Sjøberg, "Daily Stand-Up Meetings: Start Breaking the Rules", *IEEE Software*, vol. 37, str. 70–77, 2020.

[46] K. Sureshchandra, in J. Shrinivasavadhani, "Moving from waterfall to agile", *AGILE 2008*, str. 97–101. Washington DC, avgust 2008. Dostopno: https://www.researchgate.net/publication/4365409_Moving_from_Waterfall_to_Agile

[47] J. Sutherland, "Agile development: lessons learned from the first Scrum", *Cutter Agile Project Management Advisory Service: Executive Update*, vol. 5, št. 20, str. 1–4, 2004.

[48] J. Sutherland in K. Schwaber, "The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework". ScrumInc: Paris, 2011.

[49] J. Sutherland in K. Schwaber, "The Scrum Guide, The Definitive Guide to Scrum: The Rules of the Game", 2016. Dostopno: https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf

[50] H. S. Sverrisdottir, H. T. Ingason, H. I. Jonasson, "The role of the product owner in scrum - comparison between theory and practices", *Procedia - Social and Behavioral Sciences*, vol. 119, str. 257 – 267, 2014.

[51] L. Pavlič in L. Brezočnik, "Pasti in zanke agilnih pristopov", *OTS 2017 sodobne informacijske tehnologije in storitve*, str. 78–87, 13. junij 2017, Maribor, Slovenija. Maribor: Fakulteta za elektrotehniko, računalništvo in informatiko, 2017.

[52] R. Pichler, *Agile Product Management with Scrum: Creating Products that Customers Love*. New Yersey: Addison-Wesley, 2010.

[53] M. Pikkarainen, et al., "The impact of agile practices on communication in software development", *Empirical Software Engineering*, vol. 13, št. 3, str. 303–337, 2008.

[54] (2020). *Use Burn-Down Charts to Discover Scrum Anti-Patterns*. Dostopno: https://age-of-product.com/

[55] L. Williams, "What Agile Teams Think of Agile Principles", *Communications of The ACM – CACM*, vol. 55, št. 4, str. 71–76, 2012.

[56] C. Worley in E. Lawler, "Agility and Organization Design: A Diagnostic Framework". Center for Effective Organizations: Los Angeles, CA, USA, 2009. Dostopno: https://ceo.usc.edu/wp-content/uploads/2009/11/2009_12-g09-12-Agility_Org_Design.pdf