UNIVERZA V LJUBLJANI

EKONOMSKA FAKULTETA

MAGISTRSKO DELO

ANALIZA PRIMERNOSTI RAZVOJNE METODOLOGIJE INFORMACIJSKIH SISTEMOV SCRUM ZA IZBRANO RAZVOJNO EKIPO

LJUBLJANA, november 2013 DAMJAN HAUER

IZJAVA O AVTORSTVU

Spodaj podpisani Damjan Hauer, študent Ekonomske fakultete Univerze v Ljubljani, izjavljam, da sem avtor magistrskega dela z naslovom Analiza primernosti razvojne metodologije informacijskih sistemov Scrum za izbrano razvojno ekipo, pripravljenega v sodelovanju s svetovalcem prof. dr. Mirom Gradišarjem.

Izrecno izjavljam, da v skladu z določili Zakona o avtorski in sorodnih pravicah (Ur. l. RS, št. 21/1995 s spremembami) dovolim objavo magistrskega dela na fakultetnih spletnih straneh.

S svojim podpisom zagotavljam, da

- je predloženo besedilo rezultat izključno mojega lastnega raziskovalnega dela;
- je predloženo besedilo jezikovno korektno in tehnično pripravljeno v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani, kar pomeni, da sem
    - poskrbel, da so dela in mnenja drugih avtorjev oziroma avtoric, ki jih uporabljam v magistrskem delu, citirana oziroma navedena v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani, in
    - pridobil vsa dovoljenja za uporabo avtorskih del, ki so v celoti (v pisni ali grafični obliki) uporabljena v tekstu, in sem to v besedilu tudi jasno zapisal;
- se zavedam, da je plagiatorstvo – predstavljanje tujih del (v pisni ali grafični obliki) kot mojih lastnih – kaznivo po Kazenskem zakoniku (Ur. l. RS, št. 55/2008 s spremembami);
- se zavedam posledic, ki bi jih na osnovi predloženega magistrskega dela dokazano plagiatorstvo lahko predstavljalo za moj status na Ekonomski fakulteti Univerze v Ljubljani v skladu z relevantnim pravilnikom.

V Ljubljani, dne 21.11.2013 Podpis avtorja:_____

KAZALO

UVOD 1
Cilji in namen 3
Metoda dela 3
1 MANIFEST AGILNEGA RAZVOJA PROGRAMSKE OPREME 4
2 AGILNA METODOLOGIJA SCRUM 6
2.1 Scrum vloge 8
2.1.1 Skrbnik metodologije 9
2.1.2 Predstavnik naročnika 11
2.1.3 Razvojna ekipa 12
2.1.4 Vloge »piščancev« 13
2.2 Scrum proces 14
2.2.1 Sprint 15
2.2.2 Sestanek za načrtovanje sprinta 16
2.2.3 Dnevni Scrum 18
2.2.4 Revizija sprinta 19
2.2.5 Retrospektiva sprinta 19
2.3 Scrum artefakti 20
2.3.1 Seznam zahtev izdelka 20
2.3.2 Seznam zahtev sprinta 22
2.3.3 Graf preostalega časa (Burndown Chart) 23
2.3.4 Definicija »dokončanega« 26
2.4 Scrum pravila ali »pravila igre« 26
2.4.1 Sestanek za načrtovanje sprinta 26
2.4.2 Dnevni scrum 27
2.4.3 Sprint 28
2.4.4 Revizija sprinta 29
2.4.5 Retrospektiva sprinta 30
2.5 Planiranje 30
2.5.1 Zgodovinski podatki 31
2.5.2 Uporaba ekspertov 31
2.5.3 Točke uporabniških zgodb (angl.: *Story Points*) 31
2.5.4 Hitrost (angl.: *Velocity*) 32
2.5.5 Poker planiranja (angl.: *Planning Poker*) 33
3 PROCES VODENJA PROJEKTOV V OPAZOVANEM PODJETJU 35
4 OPAZOVANA RAZVOJNA EKIPA 39
5 OCENA PRIMERNOSTI METODOLOGIJE SCRUM 41
5.1 Prednosti uporabe metode Scrum 42
5.1.1 Učinkovitejše obvladovanje dokumentacije na projektu 42
5.1.2 Planiranje napora 43
5.1.3 Učinkovitost sestankov 44
5.1.4 Pretok znanja 45
5.1.5 Hitrost in učinkovitost razvoja 45
5.1.6 Primernost nove programske opreme za zahteve naročnika 46
5.1.7 Nastavljanje uteži kriterijem prednosti 47
5.2 Kriteriji primernosti razvojne ekipe za uvedbo metode Scrum 48

5.2.1 Izkušenost razvijalcev ... 48
5.2.2 Urjenje ... 48
5.2.3 Zunanja pomoč ... 49
5.2.4 Podpora vodstva ... 49
5.2.5 Organizacijska kultura ... 49
5.2.6 Sposobnost dela v timu ... 50
5.2.7 Komuniciranje ... 50
5.2.8 Razmerje z naročnikom ... 50
5.2.9 Zaznana uporabnost metode ... 50
5.2.10 Zaznana preprostost uporabe metode ... 51
5.2.11 Zaznana kompatibilnost metode ... 51
5.2.12 Prihod članov ekipe prepozno na dnevni Scrum sestanek ... 51
5.2.13 Prekinjanje dela posameznih članov z namenom, da se jih napoti na izvajanje drugih nalog izven tekočega sprinta ... 52
5.2.14 V ekipi obstajajo viri za vse zahtevane funkcionalnosti projekta (načrtovanje, oblikovanje, programiranje, testiranje, …) ... 52
5.2.15 Problem projektov s fiksno ceno in fiksnimi datumi ... 52
5.2.16 Nastavljanje uteži kriterijem primernosti ... 53
6 OCENA PRIMERNOSTI METODE SCRUM ZA OPAZOVANO EKIPO ... 54
6.1 Ocena opazovane ekipe po kriterijih prednosti uporabe metode Scrum ... 54
6.2 Ocena opazovane ekipe po kriterijih primernosti ekipe za uvedbo metode Scrum ... 58
6.3 Končna ocena ... 63
SKLEP ... 64
LITERATURA IN VIRI ... 67

KAZALO SLIK

Slika 1: Graf povezave med verjetnostjo uspeha projekta in njegovo kompleksnostjo 5
Slika 2: Scrum proces 15
Slika 3: Primer seznama zahtev izdelka 21
Slika 4: Primer seznama zahtev sprinta 23
Slika 5: Primer grafa preostalega časa objave 24
Slika 6: Primer dopolnjenega grafa preostalega časa objave 24
Slika 7: Primer grafa preostalega časa sprinta 25
Slika 8: Hitrost razvojne ekipe skozi čas 33

KAZALO TABEL

Tabela 1: Pogosto uporabljeni razredi velikosti uporabniških zgodb 32
Tabela 2: Tabela uteži za kriterije prednosti 47
Tabela 3: Tabela uteži za kriterije primernosti 53
Tabela 4: Izračun ocene prednosti uporabe metode Scrum 57
Tabela 5: Izračun ocene primernosti ekipe za vpeljavo metode Scrum 62

UVOD

Že v sredini devetdesetih let prejšnjega stoletja so se v krogih razvijalcev v svetu informacijski tehnologij pojavile in razvile agilne metodologije razvoja programskih rešitev, predvsem kot odgovor na neuspešno zadovoljevanje potreb strank z uporabo planskih (angl.: *plan-driven*) metodologij razvoja. Namen novih metodologij je bil bolje obvladovanje nenehno spreminjajočih se zahtev in pričakovanj končnih uporabnikov (strank) produktov ali storitev informacijske tehnologije (Williams, 2010, str. 3).

Tradicionalne metodologije razvoja, katerim agilne metode predstavljajo neko novo alternativo, izvirajo iz popolnoma drugačnih okolij (inženirska, večinoma iz industrije, gradbeništva in podobno), zato ni presenetljivo, da so se za potrebe projektov informacijskih tehnologij morale razviti drugače in se prilagoditi. Zahteve ciljnih strank in uporabnikov programskih rešitev niso zadovoljive zgolj s posredovanjem najboljših in tehnično najbolj izpopolnjenih rešitev, ampak je predvsem važno, da izoblikujemo rešitev, ki je najbolj prilagojena, fokusirana in naslovljena na uporabnikove specifične posebne potrebe. V tem pogledu se večina tradicionalnih metodologij ne izkaže najbolje. Pogosto se veliko truda vlaga v fazo planiranja tako, da se polovica (ali več) virov že porabi, preden se karkoli sploh razvije. Definiranje zahtev je pogosto tako delovno intenzivno in dolgotrajno, da se vmes že pojavijo nove zahteve ali pa se obstoječe že spremenijo (Cervone, 2011, str. 18).

Več avtorjev ugotavlja, da je razvoj informacijskih rešitev postala zelo zahtevna naloga. Pries in Quigley (2011, str. 1) navajata, da proces razvoja produktov postaja vse bolj zapleten in kompleksen, in da ritem tehnoloških sprememb dnevno raste in dopušča zgolj malo časa za nabiranje znanj potrebnih za začetek razvoja novega produkta.

Schwaber (1995, str. 3) pravi, da se informacijski sistemi razvijajo v visoko kompleksnih okoljih. Kompleksnost izhaja tako iz razvojnega kot ciljnega okolja. V poizkusih modeliranja takšnega procesa razvoja se srečujemo z mnogimi problemi: nekontrolirani procesi, slabo definiranje vhodov, izhodov in kontrole kakovosti (npr. procesi testiranja), uporabniške zahteve se definirajo samo v prvi fazi razvoja, kar nato zahteva kompleksne procedure upravljanja s spremembami v nadaljnjih korakih.

Klasične metodologije, kot so slapovna (angl.: *Waterfall*), spiralna in iterativna, ki so bile razvite za obvladovanje tako kompleksnega procesa razvoja, se po Schwaberjevem ugotavljanju (1995, str. 5) ne obnesejo. Slapovna metodologija je na primer preveč linearna, in ne definira dovolj dobro, kako se odzvati na nepričakovane dogodke v vmesnih procesih. Tudi izboljšave v obliki spiralne in nato iterativne metodologije razvoja se niso uspešno spopadale s problemom nepredvidljivih okolij. Strogost, ki je vpletena v razvojne procese, predstavlja preveliko oviro pri spopadanju z nepričakovanimi rezultati in kompleksnim okoljem.

Agilne metodologije so podskupina iterativnih in evolucijskih metod, in bazirajo na iterativnih izboljšavah in oportunističnih razvojnih procesih. Razvoj po agilnih metodologijah je sestavljen iz krajših iteracij, kjer vsaka predstavlja en projekt v malem z vsemi

standardnimi fazami (zajem zahtev, analiza, implementacija, testiranje in uvajanje). Produkt vsake iteracije je nova izdaja pričakovanega izdelka, ki na tak način skozi iteracije raste in preraste v končni rezultat. Agilne metodologije uporabljajo kratke iteracije, s čimer omogočajo, da se spremembe ali nove uporabniške zahteve ter drugi odzivi sproti upoštevajo in vpeljujejo v razvoj rešitve. Končni uporabniki lahko na ta način pred vsako iteracijo specificirajo nove zahteve, ki se naj vpeljejo v sledeči iteraciji razvoja, oblikovanje teh zahtev pa bazira na podlagi videnih rezultatov prejšnjih iteracij. Uporabniške zahteve tako niso zgolj špekulacije podane pred pričetkom projekta, ampak so oblikovane na podlagi videnega razvoja končne rešitve. Kratke iteracije prinašajo tudi pogoste roke, kar omogoča boljši nadzor nad potekom projekta.

Vsaka iteracija ima točno določen časovni okvir. Pred pričetkom iteracije se določi obseg dela za iteracijo tako, da bo napolnil ta časovni okvir. Tak način planiranja se razlikuje od standardnih pristopov, kjer se časovni okvir določa glede na obseg dela, ki ga želimo opraviti. V klasičnih metodah se pojavljajo iteracije dolge od 3 do 6 mesecev, medtem ko agilne metodologije uvajajo iteracije dolge od enega do štirih tednov. Raziskave so tudi pokazale, da krajše iteracije prinašajo manjša tveganja, zmanjšujejo kompleksnost, povečujejo produktivnost in večajo možnosti za uspeh (Williams, 2010, str. 3).

Tudi iz drugih virov lahko razberemo, da uvedba agilne metodologije pomeni prednost in prinaša pozitivne učinke. Raziskava iz leta 2008, ki je zajela 642 anketirancev, je pokazala, da v 86 odstotkih organizacij zaznavajo povečano produktivnost z uporabo agilnih metodologij. 77 odstotkov organizacij zaznava povečano kakovost, 78 odstotkov organizacij je zaznalo povečano zadovoljstvo interesnih skupin in v 37 odstotkih so zaznali zmanjšanje stroškov v razvoju (Ambler, 2008).

V raziskavi, ki so jo izvedli v znotraj skupine Nokia, in je zajela več kot 1000 anketirancev iz sedmih različnih držav, so prav tako ugotovili pozitivne koristi v uvedbi agilne metodologije. Med njimi večje zadovoljstvo, občutek učinkovitosti, povečanje kakovosti in transparentnosti ter zgodnejše odkrivanje napak. Prav tako se je večina udeležencev raziskave opredelila, da si ne želi vrnitve na tradicionalne metodologije (Laanti, Salo & Abrahamsson, 2010, str. 1).

Danes poznamo več metodologij, ki jih uvrščamo med agilne. Med njimi eXtreme Programming, Scrum, Dynamic Systems Development Method, Adaptive Software Development, Crystal in Feature-Driven Development (Vinekar, Slinkman & Nerur, 2006, str. 31). Neko metodologijo uvrstimo med agilne, če ustreza naslednjim osnovnim načelom agilnosti: usmerjena v ljudi, komunikacijsko orientirana, fleksibilna (prilagodljiva za nepredvidene spremembe, ki se lahko pojavijo kadarkoli), hitra (iterativen razvoj v kratkih iteracijah), vitka (»lean« - osredotočena na manjše časovne okvire in stroške z izboljšano kakovostjo), odzivna (hitro reagira na spremembe) in učna (osredotočena na izboljšave med in po razvoju) (Qumer & Henderson-Sellers, 2008, str. 281).

Kot sodelavec razvojne ekipe v podjetju, ki se ukvarja z razvojem in nudenjem storitev s področja informacijske tehnologije, se vsakodnevno srečujem s težavami, ki jih prinaša kompleksno okolje razvoja informacijskih rešitev. Iskanje rešitev za učinkovitejše delo in

boljše razvojne prijeme je tako tudi moja profesionalna naloga. Prednosti, ki jih nakazujejo agilne metodologije, bi za našo razvojno ekipo lahko predstavljale korak v pozitivno smer, saj bi lahko odpravili katero od težav oz. izboljšali kakšen del našega procesa. Predvsem bolj učinkovit razvoj rešitev, ki bi bolje zadovoljile potrebe naših strank, ter lažje obvladovanje poteka dela in boljše planiranje napora in virov. Uvedba metodologije Scrum, kot trenutno ene najbolj razširjenih agilnih metodologij, se zdi zelo mikavna ideja o kateri velja razmisliti. V okviru te naloge bi tako rad predvsem ugotovil, ali bi takšna uvedba bila izvedljiva in smotrna, ter kakšne prednosti bi s tem pridobili v naši razvojni ekipi.

Cilji in namen

Namen magistrske naloge je izboljšati proces razvoja produktov in storitev informacijske tehnologije v svoji ekipi, ki se ukvarja z razvojem poslovnih programskih rešitev. Namen želim doseči z uvedbo nove agilne metodologije Scrum, pri tem pa v nalogi predvsem oceniti, ali je uvedba takšne metodologije sploh mogoča in smotrna pri danih pogojih dela ekipe, podjetja in pri projektih, ki jih izvajamo.

Cilji naloge so:

- preučiti agilno metodologijo Scrum in prikazati razlike z obstoječo metodologijo v opazovani ekipi;
- poiskati prednosti nove metodologije in izpostaviti tiste, ki bi opazovani ekipi prinesla izboljšave;
- poiskati pogoje oz. dejavnike, ki določajo, ali je vpeljava agilne metodologije Scrum v ekipo izvedljiva in smotrna;
- podati končno oceno za opazovani primer (razvojno ekipo) o primernosti in izvedljivosti vpeljave metodologije Scrum.

Metoda dela

Pri izdelavi magistrske naloge bom uporabil znanja pridobljena na dodiplomskem in podiplomskem študiju ter praktične izkušnje pridobljene z delom na projektih informacijske tehnologije. Z vsebino obravnavanega področja se bom seznanil na naslednja načina:

- študija domače in tuje literature s področja agilnih metodologij,
- študija domačih in tujih strokovnih in znanstvenih člankov v revijah in na internetu.

Predstavitev opazovane razvojne ekipe, njenih projektov in organizacijskega okolja (podjetje) bom izvedel z lastnim opazovanjem in s pomočjo podatkov iz projektnih dokumentacij in zapisov o sistemu poslovanja v podjetju.
Teoretičen del naloge bo vseboval predstavitev agilne metodologije za razvoj informacijskih produktov imenovane Scrum, procesa Scrum metodologije, vlog, aktivnosti in artefaktov.
S preučevanjem literature bom v praktičnem delu naloge poizkusil izluščiti in izpostaviti tiste prednosti agilne metodologije Scrum, ki bi opazovani ekipi prinesle največ dodane vrednosti

in rešile probleme, s katerimi se sooča ekipa pri svojem delu. Na osnovi analize konkretnega razvojnega okolja bom z metodo sinteze določil pogoje, ki morajo biti izpolnjeni, da lahko vpeljemo metodologijo Scrum v opazovano ekipo.

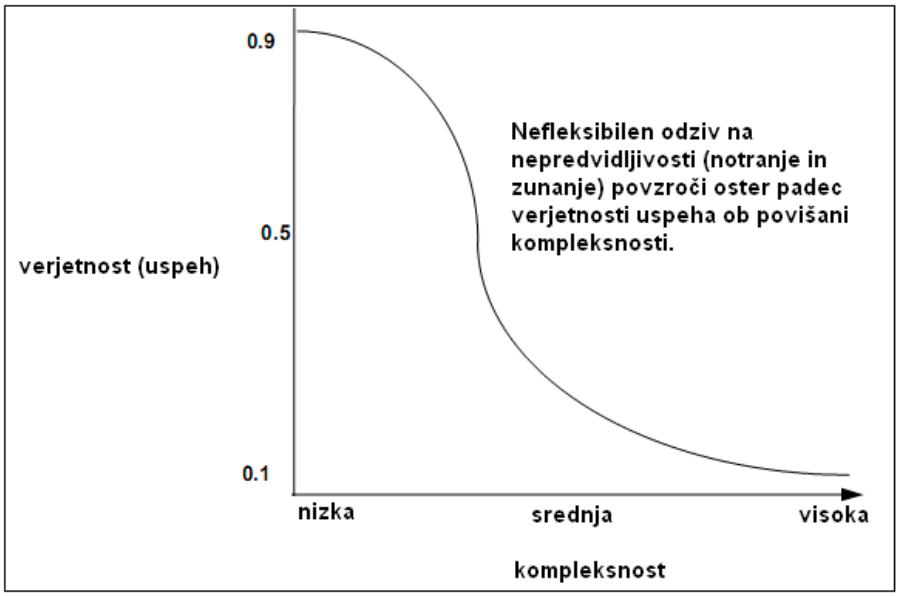
1 MANIFEST AGILNEGA RAZVOJA PROGRAMSKE OPREME

Kot sem že v uvodu omenil, so se agilne metodologije pojavile zaradi potrebe po boljšem obvladovanju kompleksnega okolja, v katerem danes nastajajo informacijske rešitve. Če želimo doseči višji uspeh na projektih, se moramo znati bolje spopasti s kompleksnostjo okolja.

Schwaber (1995, str. 8) ugotavlja, da verjetnost uspeha projekta pada z naraščanjem kompleksnosti oz. zapletenosti. Uspešnost projekta pri tem definira z uporabnostjo končne rešitve za uporabnike, kompleksnost pa kot funkcijo naslednjih spremenljivk (Schwaber, 1995, str. 3):

- Razpoložljivost izurjenih strokovnjakov: modernejša je uporabljena tehnologija, orodje ali metoda, manjši je nabor primernih razpoložljivih strokovnjakov.
- Stabilnost tehnologije izvedbe: novejša tehnologija pomeni manjšo stabilnost in večjo potrebo po uravnoteženju z drugimi tehnologijami.
- Stabilnost in moč uporabljenih orodij: novejša in močnejša orodja pomenijo manjši nabor razpoložljivih strokovnjakov in večja nestabilnost funkcionalnosti orodja.
- Učinkovitost metode: katere metode modeliranja, testiranja, upravljanja z verzijami in dizajniranja bodo uporabljene in kako so učinkovite in preizkušene?
- Domenska znanja: ali so na voljo izurjeni strokovnjaki za različne domene vključno s poslovno in tehnološko?
- Nove funkcionalnosti: koliko popolnoma novih funkcionalnosti bo dodanih in do kolikšne stopnje bodo te sovpadale z že obstoječimi funkcionalnostmi?
- Metodologija: ali izbrani pristop razvoja sistema dovolj dobro promovira fleksibilnost, ali pa gre zgolj za tog in strogo detajlni pristop, ki omejuje fleksibilnost?
- Konkurenca: kaj bo počela konkurenca v času projekta. Kakšne nove funkcionalnosti bodo najavljene ali izdane?
- Čas/finančna sredstva: koliko časa in finančnih sredstev je na voljo sprva in koliko v napredovanju projekta?
- Ostale spremenljivke: vsi ostali faktorji na katere se je, zaradi zagotavljanja uspeha nastajajočega sistema, potrebno odzivati v času projekta.

Slika 1: Graf povezave med verjetnostjo uspeha projekta in njegovo kompleksnostjo



Vir: K. Schwaber, SCRUM Development Process, 1995, str. 8.

Agilne metodologije se kompleksnega okolja pri razvoju v informacijski tehnologiji lotevajo z agilnimi pristopom. Lastnosti agilnosti se najbolje odražajo v manifestu agilnega razvoja programske opreme. Manifest je nastal februarja leta 2001 in je delo več avtorjev[1], ki se povezujejo v organizacijo »Agile Aliance« (Williams, 2010, str. 4).

Slovenski prevod manifesta agilnega razvoja programske opreme (Beck et al., 2001):

»Odkrivamo boljše načine razvoja programske opreme, tako, da jo razvijamo, in pri tem pomagamo tudi drugim. Naše vrednote so ob tem postale:

Posamezniki in interakcije pred procesi in orodji.
Delujoča programska oprema pred vseobsežno dokumentacijo.
Sodelovanje s stranko pred pogodbenimi pogajanji.
Odziv na spremembe pred togim sledenjem načrtom.

Z drugimi besedami, četudi cenimo dejavnike na desni, vseeno bolj cenimo tiste na levi.« Skupaj z manifestom so predstavili tudi 12 principov, ki definirajo agilni razvoj (Williams, 2010, str. 4; Beck et al., 2001):

- »Naša najvišja prioriteta je zadovoljiti stranko z zgodnjim in nepretrganim izdajanjem vredne programske opreme.«
  Zgodnje in ponavljajoče dostavljanje gotove programske opreme omogoča, da razvojna ekipa prejema odzive na nastajajoč izdelek že med samim razvojem. Stranka pridobi ob tem tudi večje zaupanje in ima možnost podati predloge in spremembe med razvojem, ki se tako lahko popolnoma prilagodi njenim potrebam.

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

- »Sprejemamo spremembe zahtev, celo v poznih fazah razvoja. Agilni procesi vprežejo tovrstne spremembe v prid konkurenčnosti naše stranke.«
  Agilne metodologije dopuščajo vpeljavo sprememb v zahtevah skozi ves proces razvoja, saj se projekt na novo planira ob vsaki iteraciji. Vse zahteve po spremembah so obravnavane ob vsaki novi iteraciji.
- »Delujočo programsko opremo izdajamo pogosto, znotraj obdobja nekaj tednov, do nekaj mesecev, s preferenco po krajšem časovnem okvirju.«
  Krajše iteracije pomenijo zgodnejšo in bolj pogosto dostavo nastajajoče programske opreme.
- »Poslovneži in razvijalci morajo skozi celoten projekt dnevno sodelovati.«
  Projektni vodje, menedžerji in analitiki morajo biti na voljo, da podajajo svoje odzive in za odgovore na vprašanja razvijalcev.
- »Projekte gradimo okrog motiviranih posameznikov. Omogočimo jim delovno okolje, nudimo podporo in jim zaupamo, da bodo svoje delo opravili.«
  Agilne metodologije poudarjajo pooblaščanje posameznika v ekipi, da se samostojno odloča, in da deluje v smeri dokončanja naloge in v korist celotni ekipi.
- »Najboljša in najučinkovitejša metoda posredovanja informacij razvojni ekipi in znotraj ekipe same, je pogovor iz oči v oči.«
  Ta princip postavlja sinhrono človeško komuniciranje pred komunikacijo skozi dokumente. Če osebni pogovor ni možen, so dopustni tudi drugi načini medosebne komunikacije (telefon, internetni klepet, …).
- »Delujoča programska oprema je primarno merilo napredka.«
- »Agilni procesi promovirajo trajnostni razvoj. Sponzorji, razvijalci in uporabniki morajo biti zmožni konstantnega tempa za nedoločen čas.«
  Trajnosten razvoj pomeni, da ekipa dela v takšnem ritmu, ki ga je mogoče vzdrževati skozi celoten čas projekta. Nasprotuje se prekomernemu nadurnemu delu.
- »Nenehna težnja k tehnični odličnosti in k dobremu načrtovanju izboljša agilnost.«
- »Preprostost -- umetnost zmanjševanja količine nepotrebnega dela -- je bistvena.«
  Poudarek na produktih, ki so dovolj preprosti, da z njimi obvladujemo nenehne spremembe in zadovoljujemo strankine zahteve.
- »Najboljše arhitekture, zahteve in načrti izhajajo iz tistih ekip, ki so samo-organizirane.«
- »V rednih časovnih razdobjih ekipa išče načine, kako postati učinkovitejša ob rednem prilagajanju svojega delovanja.«
  Ob vsaki iteraciji ekipa pretehta, kaj je v prejšnji iteraciji delovalo dobro in kaj slabo. Izkušnje se upoštevajo pri planiranju naslednje iteracije

2 AGILNA METODOLOGIJA SCRUM

Schwaber (2004, str. 10), soavtor metodologije Scrum (poleg Jeff Sutherland-a), v svoji knjigi »Agile Project Management with Scrum« metodologijo Scrum označi kot najbolj zmešan in paradoksen proces za upravljanje zapletenih projektov. Po eni strani lahko Scrum opišemo kot preprost, po drugi strani pa je njegova preprostost lahko varljiva. Scrum ni perspektiven proces, saj ne zna napovedati kaj storiti ob vsakršnih okoliščinah. Uporaben je

za zapletena opravila, pri katerih je nemogoče napovedati vsega, kar se lahko zgodi v nadaljevanju. Scrum ponuja okvir in zbirko praks, ki omogočajo razvojnim ekipam, da vedno dobro vedo, kaj se dogaja znotraj projekta, in da so sposobne izvesti popravke in prilagoditve takoj, ko je to potrebno, in tako ohraniti projekt, da poteka proti doseganju zastavljenega cilja (Schwaber, 2004, str. 10).

Osnovni princip Scrum metodologije lahko predstavimo na preprostem primeru izgradnje hiše. Kupec hiše oz. naročnik se ne more vseliti v novo hišo, dokler ta ni zgrajena v celoti s strani izvajalca. Če si zamislimo, da bi bil proces gradnje hiše lahko inkrementalen in iterativen, bi z uporabo takšnega pristopa hišo gradili sobo za sobo. Centralna, električna in druga napeljava bi bila vgrajena že v prvo sobo, in potem razširjena na vsako novo sobo, ki bi bila zgrajena. Kupec hiše bi se lahko vselil že takoj, ko bi se odločil, da je zgrajenih že dovolj sob za njegove osnovne potrebe. Vse dodatne sobe bi bile dograjene nato v odvisnosti od kupčevih potreb. Scrum dopušča strankam, da izgradnja programske opreme za njih poteka ravno na takšen način. Medtem, ko je osnovna infrastruktura sistema zgrajena, se koščki funkcionalnosti dostavljajo kupcu sproti, tako da ciljna organizacija lahko posamezne dele sistema uporablja že v zgodnjih fazah razvojnega cikla. Ko uporabniki že izkusijo nov sistem, se lahko tudi sproti odločajo, kateri deli sistema naj bodo dostavljeni naslednji in v kakšnem vrstnem redu. Lahko se tudi odločijo, da ne potrebujejo izgradnje celotnega sistema, če so popolnoma zadovoljni že s samo manjšim naborom funkcionalnosti (Schwaber, 2004, str. 10).

Scrum torej ni proces ali tehnika za razvoj programskih rešitev, ampak je procesni okvir znotraj katerega lahko uporabimo različne procese ali tehnike. Okvir je sestavljen iz Scrum ekip in z njimi povezanih vlog, dogodkov, artefaktov in pravil. Pravila Scruma povezujejo dogodke, vloge in artefakte, ter urejajo odnose in razmerja med njimi (Schwaber & Sutherland, 2011, str. 3).

Scrum sloni na teoriji empiričnega nadzora procesov. Ta trdi, da se znanje pridobi z izkušnjami in odločitvami na podlagi znanega (Schwaber & Sutherland, 2011, str. 3). Empirični nadzor procesov se uporabi takrat, ko definirani nadzor procesov ni mogoč zaradi zapletenosti vmesnih aktivnosti (Schwaber, 2004, str. 13).

Vsako izvršitev empiričnega nadzora procesov podpirajo trije stebri (Schwaber & Sutherland, 2011, str. 4):

- **Transparentnost**: pomembni vidiki procesa morajo biti vidni vsem, ki so odgovorni za rezultat. Za te vidike morajo biti določeni skupni standardi tako, da se opazovalci strinjajo kaj vidijo. Na primer vsi udeleženci, ko govorijo o procesih, uporabljajo skupni jezik. Tisti, ki delajo in tisti, ki sprejemajo rezultate dela, morajo deliti skupno definicijo »dokončanega«.
- **Pregled**: izvajalci morajo pogosto pregledati napredek v smeri cilja in Scrum artefakte, da zaznajo mogoča neželena odstopanja. Pregledi naj ne bodo prepogosti, da bi ovirali delo. Najbolj je, če jih na samem delovnem mestu izvajajo usposobljeni inšpektorji.

- **Prilagoditev**: če inšpektor ugotovi odstopanja izven sprejemljivih meja pri enem ali več vidikih procesa, končni izdelek pa bi bil zaradi tega nesprejemljiv, se mora prilagoditi proces. Prilagoditev se opravi čim prej, da se tako zmanjša nadaljnje odstopanje.

Značilnosti Scrum projektov (Schwaber, 1995, str. 16):

- Fleksibilnost dobave rezultatov: vsebina objavljenih izdelkov je diktirana s strani okolja in se stalno prilagaja.
- Fleksibilen urnik: objave izdelkov se lahko zahtevajo prej ali kasneje, kot je bilo prvotno planirano.
- Majhne ekipe: vsaka ekipa nima več kot 6 članov. En projekt lahko vključuje več ekip.
- Pogosti pregledi: napredek ekipe je pregledan tako pogosto kot to diktira kompleksnost okolja in tveganje (navadno 1 do 4 tedenski cikli). Funkcionalno izvedljiva objava mora biti pripravljena ob vsakem pregledu s strani vsake ekipe.
- Sodelovanje: sodelovanje znotraj ekipe in z zunanjim svetom je pričakovano med tekom projekta.
- Objektno orientirano: vsaka ekipa bo obravnavala nabor povezanih objektov z jasnimi vmesniki in obnašanji.

Prednosti Scrum metodologije (Schwaber, 1995, str. 17):

- Tradicionalne metodologije so oblikovane tako, da so se sposobne odzivati na nepredvidljivost zunanjih in razvojnih okolij zgolj ob ciklih nadgradenj. Medtem ko je Scrum metodologija oblikovana tako, da je precej prilagodljiva v celoti. Zagotavlja kontrolne mehanizme za planiranje objave nove izdaje izdelka in upravlja s spremenljivkami, tako kot napreduje projekt. To omogoča organizaciji, da spreminja projekt in izdelek ob katerikoli točki v času, in tako dosega objavo najprimernejših izdaj izdelka.
- Scrum metodologija daje razvijalcem svobodo, da lahko sami oblikujejo najbolj iznajdljive rešitve skozi celoten projekt vsakič, ko se spremeni okolje in se pojavi priložnost za učenje.
- Majhne ekipe sodelujočih razvijalcev so sposobne deliti tiho znanje o procesu razvoja. Ustvari se odlično okolje za treniranje vseh znanj znotraj ekipe.
- Objektno orientirana tehnologija predstavlja osnovo za Scrum metodologijo. Objekti oz. funkcionalnosti izdelka ponujajo diskretno in obvladljivo okolje.

2.1 Scrum vloge

Scrum ločuje ljudi na projektu med tiste, ki so projektu zavezani in na ostale, ki so samo zainteresirani v projekt in so vanj vključujejo zgolj interesno. Koncept takšnega ločevanja najlažje prikažemo s pomočjo šaljive zgodbe, v kateri nastopata prašič in piščanec. V zgodbi piščanec prašiču predlaga, da bi skupaj odprla restavracijo. Prašič ga vpraša, kako bi se restavracija imenovala, in piščanec predlaga ime »Šunka z jajci«. Prašič takšen predlog strogo zavrne in pove, da se ne strinja, saj bi bil on pri vsem preveč zavezan, medtem ko bi bil piščanec zgolj vključen (Blankenship, Bussa & Millett, 2011, str. 26). Iz te zgodbe

izhajata tudi naziva, ki po Scrum-u označujeta osebe. Zavezani v projekt se imenujejo »prašiči« (angl.: *Pigs*), medtem ko se ostali zainteresirani imenujejo »piščanci« (angl.: *Chickens*).

Prašiči so torej tiste vloge, ki so zadolžene za dejansko ustvarjanje, testiranje in uvajanje končne programske rešitve, medtem ko so piščanci tisti, ki so projektu manj zavezani in navadno predstavljajo interesne skupine, ki bodo imele koristi od izdelkov projekta a niso odgovorni za njihovo dostavo oz. proizvodnjo (Blankenship et al., 2011, str. 27).
Scrum vrednoti mnenja, predloge in druge opazke, ki prihajajo od »piščancev«, vendar ti ne smejo ovirati dela »prašičev« pri ustvarjanju izdelkov. Scrum promovira visoko podporo »prašičem« pri njihovem delu a hkrati predvideva, da je potrebno v obzir vzeti tudi poglede »piščancev« (Blankenship et al., 2011, str. 27).

Vloge, ki jih uvrščamo med »prašiče« (Pries & Quigley, 2011, str. 52):

- Skrbnik metodologije ali Scrum učitelj oz. izvirno »Scrum Master«,
- predstavnik naročnika ali lastnik izdelka oz. izvirno »Product Owner«,
- razvojna ekipa oz. izvirno »Team«.

Vloge, ki jih uvrščamo med »piščance« (Pries & Quigley, 2011, str. 53):

- Uporabniki,
- nosilci interesov (angl.: *Stakeholders*) in
- vodstvo.

V literaturi sem zasledil dva slovenska izraza za izviren naziv »Scrum Master«. Izraz »skrbnik metodologije« je uporabljal Mahnič (Mahnič, Georgiev & Jarc, 2009, str. 247), medtem ko je izraz »Scrum učitelj« uporabljen v slovenskem prevodu Scrum vodiča, ki ga najdemo na spletni strani Scrum.org. Kot avtor prevoda v slovenski jezik je naveden Jure Klofutar. Enako je tudi za naziv »Product Owner«, kjer prvi avtor (Mahnič) uporablja izraz »predstavnik naročnika«, medtem ko se v Scrum vodiču uporablja izraz »lastnik izdelka«.

2.1.1 Skrbnik metodologije

Skrbnik metodologije vodi Scrum aktivnost zlasti ob začetku projekta. Njegova primarna naloga je odpravljanje ovir pri izpolnjevanju nalog. Osebo s to vlogo ne moremo preprosto označiti kot vodjo ekipe (razen mogoče ob začetku projekta), saj se od ekipe pričakuje, da se bo znala sama organizirati. Skrbnik metodologije torej skrbi, da je Scrum proces projekta uporabljen tako, kot je bil predviden. Kot posrednik pri ohranjanju Scrum aktivnosti na nalogah se sklicuje na ustrezne postopke in pravila (Pries & Quigley, 2011, str. 52).

Za razliko od klasičnega vodje projekta skrbnik metodologije ne izdaja nalog in ne razporeja ljudi na projektu oz. jim ne narekuje, kaj naj počnejo. To po Scrum-u počnejo ekipe same, skrbnik metodologije pa jim pri tem pomaga (Sutherland, 2010, str. 16).

Skrbnika metodologije lahko označimo tudi kot promotorja Scrum metode v ekipi. Skrbeti namreč mora, da je Scrum pravilno razumljen in dobro sprejet, in da ga ekipa pravilno upošteva (se drži pravil, praks in teorije Scrum). Prav tako skrbi, da so Scrum pravila jasna tudi osebam, ki niso člani ekipe (»piščanci«) in pojasnjuje, kakšna interakcija s Scrum ekipo je lahko koristna in kakšna ne, ter pomaga, da se zagotovi zgolj takšna interakcija, ki zagotavlja maksimiranje vrednosti, ki jo ustvari Scrum ekipa (Schwaber & Sutherland, 2011, str. 6).

Ker je skrbnik metodologije ključen za zagotavljanje učinkovitega okolja razvojni ekipi, je dobro, da ima ekipa skrbnika, ki je na voljo polni delovni čas in je dobro zavezan k uspehu projekta. Pri manjših ekipah je skrbnik metodologije lahko tudi oseba iz razvojne ekipe, ki ima zato dodeljen manjši nabor nalog, da lahko igra tudi vlogo skrbnika. Nikakor pa ni primerno, da je skrbnik metodologije hkrati tudi predstavnik naročnika, saj lahko imata ti dve vlogi nasprotujoče zahteve. Predstavnik naročnika bi lahko na primer na vsak način želel uvesti novo funkcionalnost že med izvajanjem sprint-a, kar je vsekakor v neskladju s Scrum metodologijo. Tu pa mora nastopiti skrbnik metodologije, da takšne indice prepreči in uveljavi primerno Scrum pravilo (Sutherland, 2010, str. 16).

Dobri skrbniki metodologije lahko prihajajo iz različnih vlog: inženirji, preizkuševalci, oblikovalci (inf. rešitev) , vodje projektov, produktni vodje ali vodje kakovosti (Sutherland, 2010, str. 16).

Skrbnik metodologije je na voljo in v pomoč vsem, ki so vključeni v projekt.
Predstavniku naročnika služi med drugim tudi pri (Schwaber & Sutherland, 2011, str. 7):

- »Iskanju tehnik za učinkovito upravljanje s seznamom zahtev izdelka.«
- »Jasnem izražanju vizije, ciljev in predmetov s seznama zahtev izdelka razvojni ekipi.«
- »Poučevanju razvojne ekipe kako ustvariti jasne in jedrnate predmete na seznamu zahtev izdelka.«
- »Razumevanju dolgoročnega načrtovanja v empiričnem okolju.«
- »Razumevanju in prakticiranju agilnosti.«
- »Omogočanju Scrum dogodkov kot je zahtevano oziroma potrebno.«

Razvojni ekipi služi med drugim tudi pri (Schwaber & Sutherland, 2011, str. 7):

- »Poučevanju razvojne ekipe na področju samo-organizacije in navzkrižne funkcionalnosti.«
- »Poučevanju in vodenju razvojne ekipe pri ustvarjanju visoko kakovostnih izdelkov.«
- »Odstranjevanju ovir pri napredku razvojne ekipe.«
- »Omogočanju Scrum dogodkov kot je zahtevano oziroma potrebno.«
- »Poučevanju razvojne ekipe v organizacijskem okolju, kjer Scrum še ni popolnoma uveden in razumljen.«

Organizaciji služi med drugim tudi pri (Schwaber & Sutherland, 2011, str. 7):

- »Vodenju in poučevanju organizacije pri uvajanju Scruma.«

- »Načrtovanju izvršitve Scruma v organizaciji.«
- »Pomoči zaposlenim in nosilcem interesov pri razumevanju ter sprejemanju Scruma in empiričnega razvoja izdelkov.«
- »Povzročanju sprememb, ki povečujejo produktivnost Scrum ekipe.«
- »Sodelovanju z ostalimi skrbniki metodologije pri povečevanju uspešnosti uporabe Scruma v organizaciji.«

Če razvojna ekipa predstavlja motor Scrum projekta, potem si skrbnika metodologije lahko predstavljamo kot olje, ki skrbi, da motor teče nemoteno (Blankenship et al., 2011, str. 27).

2.1.2 Predstavnik naročnika

Predstavnik naročnika predstavlja glas stranke oz. naročnika programske rešitve. Oseba s to vlogo preverja, če je delo ekipe učinkovito v poslovnem smislu (Pries & Quigley, 2011, str. 77). Odgovoren je za maksimiranje vrednosti produkta, ki ga izdela razvojna ekipa. Srečuje se z naročniki in ugotavlja njihove želje in potrebe in postavlja prioritete nad nalogami tako, da razvojna ekipa vedno dela na tistih točkah, ki za naročnika v danem trenutku predstavljajo največjo vrednost. Upravlja seznam zahtev izdelka in je edina oseba, ki lahko nastavlja prioritete uporabniških zgodb za naslednji sprint. Vse funkcionalnosti so v bistvu razvite za osebo z vlogo predstavnika naročnika. Predstavnik naročnika je tudi odgovoren za potrditev rezultatov sprinta.

Njegove odgovornosti na projektu se spreminjajo in ga lahko klasificiramo kot »prašiča« pred in po vsakem sprintu, medtem ko je v času izvajanja sprint-a uvrščen med »piščance«. Scrum ekipa ne more delovati brez predstavnika naročnika. Slab predstavnik naročnika, ki ne zmore natančno prikazati potreb in želja strank, ter zagotoviti, da bo izdelana programska oprema predstavljala visoko vrednost za naročnika, lahko povzroči neuspeh projekta (Blankenship et al., 2011, str. 27).

Želene lastnosti dobrega predstavnika naročnika (Pichler, 2010, str. 3):

- **Vizionar in storilec**: vizionar, ki lahko predvidi končen produkt in zna predstaviti svojo vizijo o njem ter storilec, ki vzdržuje vizijo do zaključka. To vključuje opisovanje zahtev, tesno sodelovanje z razvojno ekipo in krmiljenje projekta s spremljanjem in napovedovanjem napredka. Spodbuja kreativnost in inovativnost ter je dojemljiv za spremembe, ambicije, debate, konflikte, eksperimentiranje in sprejemanje tveganja.
- **Vodja in član ekipe hkrati**: vodja odgovoren za uspeh produkta, ki lahko določa smernice za vsakogar, ki je vključen v razvoj. Član ekipe, ki tesno sodeluje z ekipo vendar nima formalne avtoritete nad njenim delom. Diktiranje odločitev ni dobra praksa, prav tako ne neodločnost predstavnika naročnika. Odločitve sprejema v konsenzu z ekipo, pri tem pa se spodbuja in izrablja njeno kreativnost in znanje.
- **Komunikator in pogajalec**: učinkovit komunikator in pogajalec, ki komunicira in usklajuje različne vloge na projektu vključujoč naročnike, uporabnike, razvijalce, inženirje, marketing, prodajalce in menedžment.

- **Usposobljen in predan**: imeti mora dovolj avtoritete in pravo mero podpore vodstva, da lahko usklajuje prizadevanja razvoja z interesnimi skupinami. Usposobljen, da vodi prizadevanja, ki pripeljejo do končnega produkta in s pravilnimi odločitvami določa, katere funkcionalnosti bodo dostavljene v posamezni izdaji.
- **Dostopen in kvalificiran**: zadolžitve predstavnika naročnika navadno zahtevajo človeka za polni delovni čas. Zahtevana je kvalificiranost za podrobno razumevanje strank, trga, uporabniških izkušenj in zmožnost dogovarjanja o potrebah in zahtevah uporabnikov ter sodelovanja s samo-organizirano razvojno ekipo.

Predstavnik naročnika je edina oseba, ki je odgovorna za upravljanje seznama zahtev izdelka. Ta naloga vključuje (Schwaber & Sutherland, 2011, str. 5):

- »Jasno opredelitev predmetov s seznama zahtev izdelka.«
- »Razvrščanje predmetov s seznama zahtev izdelka, tako da se kar najbolje doseže cilje in poslanstva.«
- »Zagotavljanje vrednosti dela, ki ga opravlja razvojna ekipa.«
- »Zagotavljanje, da je seznam zahtev izdelka vsem viden, pregleden in razumljiv, ter da kaže prihodnje delo Scrum ekipe.«
- »Zagotavlja, da razvojna ekipa dovolj dobro razume predmete s seznama zahtev izdelka.«

Zgoraj opisano delo lahko opravlja sam, ali pa to počne razvojna ekipa, vendar mora odgovornost za seznam zahtev izdelka vedno nositi predstavnik naročnika.

Vlogo predstavnika naročnika nosi vedno le ena oseba, ki pa lahko predstavlja želje širšega odbora glede seznama zahtev izdelka, pri tem se nobena sprememba na seznamu zahtev izdelka ne zgodi brez privolitve predstavnika naročnika.

Odločitve predstavnika naročnika morajo biti spoštovane in upoštevane s strani celotne organizacije in so vidne v vsebini in prioritetah na seznamu zahtev izdelka. Razvojna ekipa lahko dela samo na nalogah iz seznama zahtev izdelka, in nihče drug ji ne more naročiti, da razvijajo druge zahteve ali v drugačnem vrstnem redu (Schwaber & Sutherland, 2011, str. 5).

V nekaterih primerih je lahko predstavnik naročnika tudi naročnik sam (ista oseba), na primer pri notranjih projektih razvoja programskih rešitev za potrebe lastne organizacije (Sutherland, 2010, str. 14).

2.1.3 Razvojna ekipa

Razvojna ekipa je skupina oseb, ki so odgovorne, da dejansko razvijejo končno programsko rešitev. Vključuje različne profile ljudi od programerjev, preizkuševalcev in oblikovalcev uporabniških vmesnikov do oseb s še drugimi znanji in spretnostmi, ki so potrebne na projektu. Ključna lastnost Scrum razvojne ekipe je njena samo-organiziranost. Nihče v ekipi ni vodja in vsi delujejo kot skupina zavezana k uspehu vsakega sprinta. Razvojne ekipe so oblikovane tako, da optimizirajo fleksibilnost in produktivnost. Od vseh članov ekipe se pričakuje, da so spoznani z vsemi aspekti produkta. Vsakdo seveda ni ekspert iz vseh iskanih področij, vendar je z vsem približno seznanjen. Razvojna ekipa sodeluje s skrbnikom

metodologije in predstavnikom naročnika z namenom izpolnitve vseh uporabniških zgodb (angl.: *User Stories*) in uspešnim zaključkom vsakega sprinta. Skrbnik metodologije je postavljen, da skrbi za interese razvojne ekipe, predstavnik naročnika pa za interese stranke. Z njima na mestu se lahko razvojna ekipa popolnoma osredotoči na izdelavo programske opreme, ki jo potrebuje končni naročnik (Blankenship et al., 2011, str. 22).

Schwaber in Sutherland (2011, str. 6) navajata naslednje značilnosti Scrum razvojnih ekip:

- »Se samo-organizirajo. Nihče (niti predstavnik metodologije) ne določa razvojni ekipi kako uporabiti seznam zahtev izdelka za ustvarjanje razširitve funkcionalnosti, ki je potencialno pripravljena za izdajo.«
- »Razvojne ekipe so navzkrižno funkcionalne; imajo vse sposobnosti, ki so potrebne za ustvarjanje razširitve izdelka.«
- »Scrum metodologija ne definira nobenega drugega naziva za člane razvojne ekipe razen naziva »razvijalec«. Ne glede na delo, ki ga posameznik opravlja, nosijo vsi člani razvojne ekipe naziv »razvijalec«.«
- »Posamezni člani razvojne ekipe imajo lahko specializirane sposobnosti in področja na katera se osredotočajo, vendar pa je razvojna ekipa odgovorna kot celota.«
- »Razvojne ekipe ne vključujejo skupin, ki bi bile namenjene posebnim področjem npr. testiranju ali poslovni analizi.«

Velikost razvojne ekipe naj bo določena glede na količino potrebnega dela. Prevelike ekipe niso primerne, saj ne morejo biti dovolj agilne. V premajhnih ekipah se lahko zmanjša interakcija med člani in posledično pade produktivnost. Prav tako lahko v majhnih ekipah prihaja do pomanjkanja določenih znanj in omejevanja v sposobnostih, ki so potrebna za izdelavo razširitve za izdajo nove programske opreme. Več kot devet članov zahteva preveč koordinacije in preveliko kompleksnost za upravljanje empiričnega procesa. V velikost ekipe se ne prištevata skrbnik metodologije in predstavnik naročnika, razen če tudi kdo od njiju dela na realizaciji točk s seznama zahtev izdelka (Schwaber & Sutherland, 2011, str. 6).

Najprimernejše velikosti razvojnih ekip so navedene različno v literaturi. Schwaber in Sutherland (2011, str. 6) predlagata 3 do 9 članov, Blankenship, Bussa in Millett (2011, str. 22) predlagajo od 2 do 10 članov, Sutherland (2010, str. 15) navaja 7 +/- 2 člana ter Pries in Quigley (2011, str. 53) predlagata 5 do devet članov. Pri sestavi ekipe in določanju njene velikosti se je torej najbolje držati priporočil iz prejšnjega odstavka.

Za najboljšo uspešnost razvojne ekipe je bistveno, da je ekipa sto odstotno zavezana k delu na sprint-u enega projekta, delo na več projektih hkrati ni zaželeno. Prav tako lahko menjave članov ekipe (dodajanje novih ali odvzemanje obstoječih članov) pripeljejo do nestabilne ekipe in zmanjšanja produktivnosti (Sutherland, 2010, str. 15).

2.1.4 Vloge »piščancev«

Med vloge, ki niso zavezane k razvoju končne rešitve imajo pa pri tem interes, štejemo:

- končne uporabnike, ki so individualne osebe za katere se razvija programska oprema;

- interesne skupine, ki predstavljajo ljudi na katere bo projekt imel kakršenkoli vpliv (npr. stranke, dobavitelji, notranje stranke, …);
- menedžment, ki je navadno lastnik procesa in okolja v katerem ekipa funkcionira (Pries & Quigley, 2011, str. 53).

Skrbnik metodologije mora poskrbeti, da »piščanci« ne delujejo na razvojno ekipo kot moteč element, vendar mora njihov vpliv jemati v obzir. Navadno se jih lahko povabi na določene Scrum sestanke kot opazovalce dogajanja (Blankenship et al., 2011, str. 22).

2.2 Scrum proces

V nadaljevanju bom opisal, kako poteka delo po Scrum metodi in kateri so glavni dogodki, ki jih definira metoda Scrum.

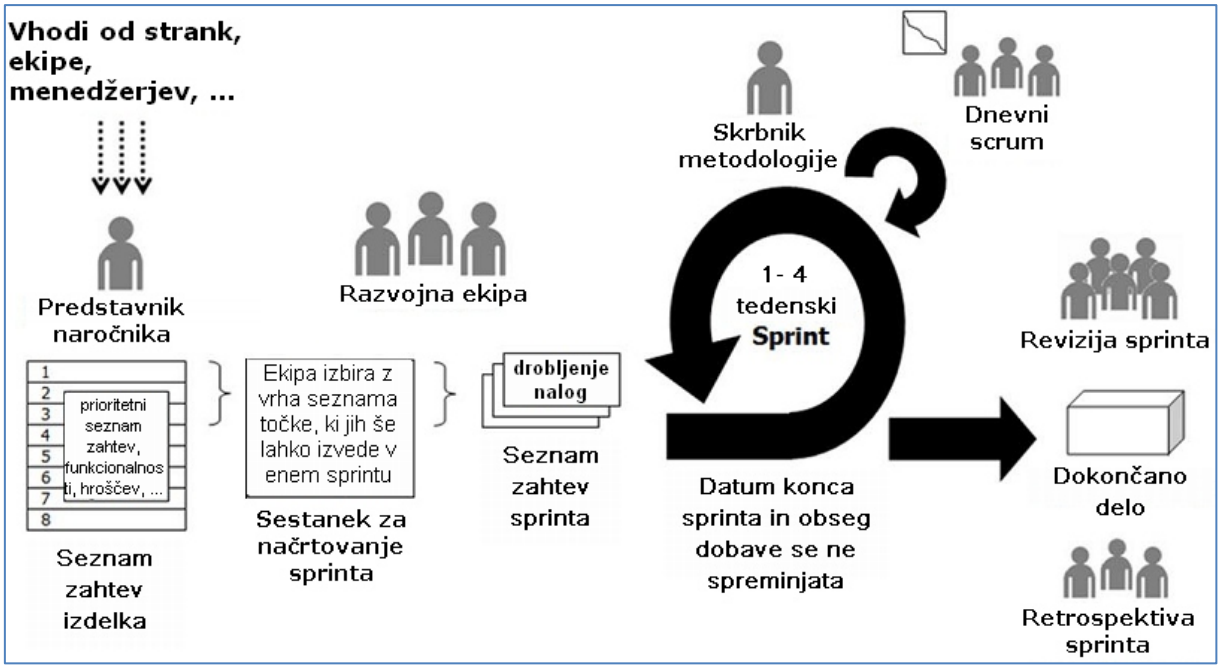
Scrum predpisuje naslednje dogodke v procesu izdelave končne programske opreme:

- sestanek za načrtovanje sprinta (angl.: *Sprint Planning Meeting*),
- sprint,
- dnevni Scrum (angl.: *Daily Scrum Meeting*),
- revizija sprinta (angl.: *Sprint Review Meeting*),
- retrospektiva sprinta (angl.: *Sprint Retrospective Meeting).*

Scrum dogodki zagotavljajo stalnost na projektu (nemoteno in neprekinjeno delo) in zmanjšujejo potrebo po nenapovedanih sestankih (tistih, ki jih Scrum ne določa). Vsi dogodki so časovno omejeni oz. imajo določen časoven okvir oz. maksimalno časovno okno.

Vsak dogodek v Scrumu je priložnost za pregled in prilagoditve, skupaj pa zagotavljajo transparentnost in pregled. Izpuščanje dogodkov ni dovoljeno, saj se s tem zmanjšuje transparentnost in se izgubljajo priložnosti za pregled in prilagoditev (Schwaber & Sutherland, 2011, str. 7).

Slika 2: Scrum proces



Vir: K. Olson, HEI Embraces Scrum Project Management, 2010, str. 45.

Scrum projekt se začne z vizijo, ki predstavlja idejo programske opreme, ki se naj izdela. Vizija je na začetku lahko nejasna, izražena bolj s tržno terminologijo, a bo postala bolj jasna, ko bo projekt napredoval. Predstavnik naročnika je tisti, ki pripravi vizijo na način, da ta najbolje maksimira donosnost. Izdela tudi plan za izpolnitev vizije, ki ga formulira v obliki seznama zahtev izdelka. Seznam zahtev izdelka je seznam funkcionalnih in nefunkcionalnih zahtev, ki bodo izpolnile vizijo, ko bodo razvite v končne funkcionalnosti (Schwaber, 2004, str. 20).

2.2.1 Sprint

Pri Scrum-u se vso delo projekta opravi v ciklih imenovanih »Sprint«. Vsak sprint je ena iteracija 30-ih zaporednih dni (Schwaber, 2004, str. 20). Sprint je lahko tudi krajši od enega meseca (pri manjših projektih), vendar v času razvoja vedno vsak sprint traja enako dolgo. Sprint predstavlja časovno okno v katerem je izdelana različica programske opreme, ki je uporabna in potencialno pripravljena na izdajo.

Sprint se prične s sestankom za načrtovanje sprinta in vsebuje še dnevne Scrum-e, razvoj, revizijo sprinta in retrospektivo sprinta (Schwaber & Sutherland, 2011, str. 8).

Med sprintom (Schwaber & Sutherland, 2011, str. 8):

- »Se ne uvaja sprememb, ki bi vplivale na cilj.«
- »Sestava razvojne ekipe ostaja konstantna.«
- »Cilji glede kvalitete se ne zmanjšujejo.«

- »Predstavnik naročnika in razvojna ekipa se lahko ponovno pogajata o obsegu dela, ko postanejo zahteve bolj jasne.«

Vsak sprint si lahko predstavljamo kot projekt v malem s katerim želimo nekaj doseči v točno določenem časovnem okviru, ki ne sme biti daljši od enega meseca. Sprint tako vsebuje definicijo končnega izdelka, fleksibilen načrt izdelave, opravljeno delo ter končni produkt.

Najdaljši sprint je lahko dolg en mesec. S predolgimi sprinti se poveča kompleksnost in tveganje, prav tako pa se lahko preveč spremeni definicija končnega izdelka. Tako časovno omejeni sprinti zagotavljajo, da se lahko najmanj enkrat mesečno izvedejo pregledi in prilagoditve napredka proti cilju. Sprint-i omogočajo predvidljivost in omejijo stroškovno tveganje na en koledarski mesec.

V redkih in izjemnih primerih se lahko sprint prekliče tudi pred iztekom njegovega časovnega okvirja. To avtoriteto ima samo predstavnik naročnika. Preklic sprinta se zgodi kadar cilj sprinta postane preveč zastarel in sprinta ni več smotrno opravljati (če, na primer, podjetje spremeni usmeritev ali pa se spremenita trg ali tehnologija).

Pri preklicu sprinta se pregleda vse že zaključene naloge in se izda nova programska oprema, če je kakšna nova funkcionalnost primerna za izdajo. Vse nedokončane naloge s seznama zahtev izdelka se ponovno ocenijo in vrnejo na seznam. Takšni drastični posegi v delo ekipe so preveč travmatični in porabljajo vire ter čas na projektu (ponovno sestajanje in načrtovanje), zato se zgodijo zelo redko (Schwaber & Sutherland, 2011, str. 8).

Eden temeljnih načel Scrum-a je tudi ta, da se trajanje sprinta nikoli ne podaljšuje. Sprint se vedno (razen ob izrednih prekinitvah) konča na dogovorjen dan, ne glede na to, če ekipa ni uspela doseči zastavljene zaveze. Razvojna ekipa se navadno ob prvih sprintih zaveže za preveliko količino opravljenega dela, kot ga dejansko nato uspe izvesti v času sprinta. Šele ob tretjem ali četrtem sprintu razvojna ekipa dobi pravi občutek za oceno lastne sposobnosti in hitrosti, da lahko pravilno zastavi obseg dela za en sprint. Takšna konstantna dolžina sprinta omogoča, da se razvojna ekipa uči bolje planirati delo in ohranja ritem delovanja razvojne ekipe (Sutherland, 2010, str. 27).

2.2.2 Sestanek za načrtovanje sprinta

Ob začetku vsakega sprinta razvojna ekipa s predstavnikom naročnika izvede sestanek za načrtovanje sprinta. Na tem sestanku se ekipa pogaja, katere točke seznama zahtev izdelka nameravajo izvesti in jih implementirati v delavni produkt ob koncu sprinta. Predstavnik naročnika je zadolžen, da napove, katere točke seznama zahtev izdelka so najpomembnejše za naročnika (imajo zanj najvišjo funkcionalno vrednost), ekipa pa poda oceno napora za implementacijo vsake predlagane točke. Izbrane točke s seznama zahtev izdelka se tako prenesejo na seznam zahtev sprinta (James, 2010, str. 2). Ekipa mora iz točk, ki jih predstavnik naročnika predstavi kot najbolj zaželene, izbrati točno takšen (toliko velik) nabor zahtev, za katerega verjame, da ga lahko implementira v delujoče funkcionalnosti v enem sprintu.

Sestanek za načrtovanje sprinta ne sme trajati dlje od osmih ur. Takšen je največji dovoljeni časovni okvir, ki tako preprečuje, da bi se porabilo preveč časa za diskutiranje o tem, kaj je mogoče izdelati. Cilj je, da se pristopi k delu, in ne, da se razmišlja o delu (Schwaber, 2004, str. 20).

Časovni okvir osmih ur je predviden za enomesečne sprinte. Za krajše sprinte se časovni okvir sorazmerno skrajša. Na primer za dvotedenski sprint se sestanek za načrtovanje sprinta omeji na štiri urni časovni okvir (Schwaber & Sutherland, 2011, str. 9).

Sestanek za načrtovanje sprinta je razdeljen v dva dela. Pri tem je za vsak del namenjena točno polovica časa celotnega sestanka.

V prvem delu sestanka se določi, kaj bo narejeno v tekočem sprintu. Predstavnik naročnika predstavi najbolj želene točke s seznama zahtev izdelka, celotna ekipa pa zagotovi, da so točke vsem razumljive (Schwaber & Sutherland, 2011, str. 9). Razvojna ekipa v tem delu predstavnika naročnika sprašuje o vsebini, namenu, pomenu in namerah točk s seznama zahtev izdelka (Schwaber, 2004, str. 20). Razvojna ekipa nato določi količino točk seznama zahtev produkta, ki jih lahko izvede v tekočem sprintu. Razvojna ekipa je edina, ki lahko poda oceno, kaj lahko doseže v tekočem sprintu. Izdela se tudi cilj sprinta, ki ga razvojna ekipa želi doseči med izvajanjem sprinta z izvršitvijo zahtev izdelka in razvojni ekipi ponuja motivacijo za ustvarjanje prirastka (Schwaber & Sutherland, 2011, str. 9).

Razvojna ekipa in predstavnik naročnika tudi dorečeta definicijo »dokončanega« (angl.: *Definition of Done*), ki mora biti v času sprinta dosežena na vseh izbranih točkah. Definicija dokončanega mora biti usklajena med vsemi člani ekipe in jo morajo vsi enako razumeti. S to definicijo se določi, kaj vse mora biti doseženo, da lahko eno točko seznama opredelimo kot dokončano (npr. programska koda je usklajena s sprejetim standardom kodiranja, koda je pregledana, testne procedure so napisane, koda je dokumentirana ipd.) (Sutherland, 2010, str. 20).

V drugem delu sestanka za načrtovanje sprinta se doreče, kako bo izbrano delo dokončano. Nabor izbranih točk s seznama zahtev izdelka se skupaj z načrtom izvedbe zabeleži v seznam zahtev sprinta (Schwaber & Sutherland, 2011, str. 9). Pri tem se upošteva hitrost izpolnjevanja nalog iz prejšnjih sprintov in razpoložljivost ter zmogljivost razvojne ekipe (Sutherland, 2010, str. 21). S tem si razvojna ekipa sama izdela podroben načrt za izvedbo sprinta, saj je samo-orgnaizirana razvojna ekipa sama odgovorna za upravljanje z delom (Schwaber, 2004, str. 20).

Na koncu sestanka bo razvojna ekipa imela pripravljen seznam nalog z ocenami, ki pa predstavlja zgolj začetno stanje, saj se bo seznam še dopolnjeval med samim sprintom. Vrstni red točk na seznamu zahtev sprinta si razvojna ekipa postavi sama tako, da se doseže najvišja hitrost in produktivnost dela (Sutherland, 2010, str. 21). Predstavnik naročnika načeloma v tem drugem delu sestanka za načrtovanje sprinta ni potreben, lahko pa je prisoten za kakšne dodatne pojasnitve glede zahtev. Lahko se tudi zgodi, da razvojna ekipa ugotovi, da je nabor točk prevelik ali premajhen za en sprint, in se v tem primeru lahko zopet pogaja s predstavnikom naročnika o naboru točk. Razvojna ekipa lahko na sestanek povabi tudi druge

osebe, če meni, da lahko kako pomagajo s tehničnimi nasveti z različnih področij (Schwaber & Sutherland, 2011, str. 9).

2.2.3 Dnevni Scrum

Ko se je izvajanje sprinta že začelo, razvojna ekipa opravlja še eno izmed ključnih Scrum praks: dnevni Scrum. Gre za kratek (15 minut ali manj) sestanek, ki se zgodi vsak delovni dan ob dogovorjenem času in na dogovorjenem mestu. Sestanka se udeležijo vsi člani razvojne ekipe. Priporočeno je, da udeleženci med sestankom stojijo, zaradi česar sestanek ne bo trajal predolgo. Dnevni Scrum je priložnost, da člani razvojne ekipe drug drugemu poročajo in pregledujejo napredek in ovire pri delu (Sutherland, 2010, str. 25).

Na sestanku vsak član razvojne ekipe po vrsti odgovori na tri vprašanja (Schwaber, 2004, str. 20):

- Kaj si uspel narediti na projektu od zadnjega dnevnega Scrum sestanka?
- Kaj nameravaš storiti na projektu od sedaj pa do naslednjega dnevnega Scrum sestanka?
- S kakšnimi ovirami se srečuješ pri uresničevanju svojih obveznosti na tekočem sprintu in projektu?

Dnevni Scrum služi kot orodje samo-organizirani ekipi in ni mišljen kot poročanje vodstvu. Je čas za deljenje občutkov med člani ekipe, kar pomaga pri koordiniranju in optimiziranju dela. Na sestanku naj nekdo beleži zapiske o opaženih ovirah, skrbnik metodologije pa je odgovoren, da pomaga razvojni ekipi pri odpravljanju teh ovir. Med dnevnim Scrum sestankom naj se opravlja samo poročanje, to ni prostor za izvajanje diskusij. Če je potreba po diskusiji, naj se ta opravi na posebnem sestanku takoj po dnevnem Scrum-u. Takšen poseben sestanek je kar pogost dogodek, kjer se razvojna ekipa opredeljuje glede informacij, ki so bile slišane na dnevnem Scrum-u. V splošnem je priporočeno, da se dnevnih Scrum sestankov vodstvo ne udeležuje, saj bi lahko člani razvojne ekipe čutili pritisk avtoritete oz. preverjanja, kar bi lahko prineslo slabše poročanje o problemih in napačno poročanje o napredku ter onemogočalo dobro samo-organiziranost razvojne ekipe (Sutherland, 2010, str. 25).

Razvojna ekipa dnevni Scrum uporablja tudi za ocenjevanje napredka v doseganju cilja sprinta in napredka v smeri izpolnitve seznama zahtev sprinta. S tem se optimizira verjetnost, da bo dosežen cilj sprinta. Dnevni Scrum razvojni ekipi tudi omogoča, da je vsak dan sposobna predstavniku naročnika in skrbniku metodologije prikazati, kako namerava v preostanku sprinta delovati in dosegati cilje sprinta. Vloga skrbnika metodologije je, da poskrbi, da se dnevni Scrum zares izvaja vsakodnevno in pomaga razvojni ekipi, da sestanek obdrži znotraj 15 minutnega časovnega okvira. Izvajanje dnevnih Scrum sestankov prinaša precej prednosti: izboljšuje komunikacijo, izniči potrebo po drugih sestankih, prepozna in odstrani ovire pri razvoju, spodbudi hitro sprejemanje odločitev in izboljša stopnjo znanja razvojne ekipe (Schwaber & Sutherland, 2011, str. 10).

2.2.4 Revizija sprinta

Ob koncu sprinta se zgodi sestanek, ki ga imenujemo revizija sprinta. Za ta sestanek je predvideno trajanje največ štiri ure (Schwaber, 2004, str. 20). Tu razvojna ekipa s predstavnikom naročnika pregleda opravljen sprint. Glavni ideji tega dogodka sta pregled in prilagoditve produkta oz. programske opreme, ki se razvija. Predstavnik naročnika in druge interesne skupine se lahko seznanijo, kaj se dogaja s produktom in delom razvojne ekipe. Prav tako pa se ob tej priložnosti razvojna ekipa seznanja z delom predstavnika naročnika (ugotovljene nove zahteve ipd.). Pričakuje se poglobljena debata med predstavnikom naročnika in člani razvojne ekipe. Revizija sprinta vključuje tudi prikaz novih funkcionalnosti, ki so bile dograjene tekom sprinta. Pri tem se ne pričakuje kakšna posebna predstavitev, zgolj prikaz novih stvari na programski rešitvi. Prisostvujejo skrbnik metodologije, predstavnik naročnika, razvojna ekipa ter zunanji zainteresirani (naročnik, stranka, uporabniki, …). Za ta del revizije sprinta je predvideno 2 urno časovno okno (Sutherland, 2010, str. 28).

Schwaber in Sutherland (2011, str. 11) navajata naslednje elemente revizije sprinta:

- Predstavnik naročnika ugotovi, kaj je bilo »dokončano« in kaj ni bilo »dokončano«.
- Razvojna ekipa razpravlja o stvareh, ki so v sprintu potekale dobro, o problemih na katere je naletela in kako so bili ti problemi rešeni.
- Razvojna ekipa predstavi »dokončano« delo in odgovori na vprašanja glede razširitve.
- Predstavnik naročnika predstavi trenutno stanje seznama zahtev izdelka. On ali ona predvidi verjetne roke za dokončanje glede na dosedanji napredek.
- Vsi skupaj sodelujejo pri odločanju o tem kaj narediti v prihodnje s čimer revizija sprinta poskrbi za dragocen prispevek k poznejšemu načrtovanju sprinta.

Revizija sprinta je lahko tudi krajša od štirih ur, če imamo primerno krajše sprinte. Kot rezultat revizije sprinta se šteje pregledan seznam zahtev izdelka, ki je prilagojen novim priložnostim in definira, kateri so verjetni predmeti za naslednji sprint (Schwaber & Sutherland, 2011, str. 11).

2.2.5 Retrospektiva sprinta

Po reviziji sprinta in pred sestankom za načrtovanje sprinta skrbnik metodologije skupaj z razvojno ekipo izvede še sestanek retrospektive sprinta. Na tem tri-urnem sestanku skrbnik metodologije poskuša spodbuditi razvojno ekipo, da revidira proces in prakse razvoja zadnjega sprinta in poskuša poiskati izboljšave in narediti proces bolj učinkovit in prijeten (Schwaber, 2004, str. 20).

Tako kot je revizija sprinta namenjena pregledu in prilagoditvi produkta, je ta sestanek namenjen pregledu in prilagoditvi procesa. Izvedba tega sestanka je zelo priporočljiva, saj dobra samo-organiziranost ekipe zahteva pogoste in redne vpoglede v funkcioniranje ekipe. Retrospektiva je priložnost za izboljšave, ki jo prinaša preglednost Scrum procesa. Celotni

Scrum ekipi omogoča diskusijo o praksah, ki delujejo in praksah, ki ne delujejo, ter diskusijo o usklajevanju sprememb (Sutherland, 2010, str. 28).

Nastane lahko načrt izboljšav, ki se uvedejo v naslednjem sprintu. Čas sestanka se zopet primerno skrajša za krajše sprinte. Vloga skrbnika metodologije je, da spodbuja razvojno ekipo k iskanju izboljšav za svoj razvojni proces in prakse. Načrtujejo se načini za izboljšavo kvalitete izdelka in se primerno prilagodi definicija »dokončanega«. Izboljšave v delovanju razvojne ekipe se načeloma lahko uvajajo kadarkoli, a je retrospektiva sprinta tisti dogodek, ki ga metoda Scrum definira ravno za to, da se ekipa vsaj takrat osredotoči na pregled in prilagoditve procesa (Schwaber & Sutherland, 2011, str. 11).

Tri glavne namene retrospektive sprinta navajata Schwaber in Sutherland (2011, str. 11):

- Pregledati potek zadnjega sprinta v zvezi z ljudmi, odnosi, procesi in orodji.
- Prepoznati in po vrsti urediti glavne zadeve, ki so šle dobro in potencialne izboljšave.
- Izdelati načrt za uvedbo izboljšav v način dela Scrum ekipe.

2.3 Scrum artefakti

Scrum vpeljuje tri glavne artefakte, ki nastajajo kot stranski produkt Scrum aktivnosti. Ti artefakti so seznam zahtev izdelka, seznam zahtev sprinta in graf preostalega dela. Med artefakte lahko štejemo še definicijo »dokončanega« kot nek stranski artefakt Scrum-a. Artefakti Scrum ekipi dajejo splošne usmeritve in zagotavljajo transparentnost (Blankenship et al., 2011, str. 16).
Artefakti predstavljajo delo in vrednost na različne načine in dajejo možnost za pregled in prilagoditev. So specifično oblikovani za maksimiranje transparentnosti ključnih informacij, potrebnih za uspešnost Scrum ekipe (Schwaber & Sutherland, 2011, str. 12).

2.3.1 Seznam zahtev izdelka

Zahteve programske opreme, izdelka ali informacijske rešitve, ki naj bo razvita, so naštete v seznamu zahtev izdelka. Predstavnik naročnika je odgovoren za vsebino, prioritete in razpoložljivost seznama zahtev izdelka. Seznam zahtev izdelka ni nikoli dokončan in se nenehno razvija, kakor se razvija okolje v katerem je uporabljen. Gre za dinamičen seznam, ki ga nenehno spreminjamo, da bi z njim identificirali, kaj je potrebno narediti na produktu, da bo ta primeren, konkurenčen in uporaben. Dokler obstaja produkt oz. izdelek projekta, obstaja tudi seznam zahtev izdelka (Schwaber, 2004, str. 22).

Seznam zahtev izdelka je enoten pogled na vse, kar je lahko kadarkoli narejeno s strani ekipe po prioritetnem vrstnem redu. Obstaja vedno samo en seznam zahtev izdelka, ki primarno vsebuje vse zahteve stranke, zraven pa tudi cilje razvojnih izboljšav (npr. zahteva za »refactoring« kode ipd.), raziskovalno delo, performančne in varnostne zahteve in znane napake (če napak ni preveč, in zato ni vzpostavljen kakšen drug sistem beleženja napak).

Predstavnik naročnika nenehno posodablja seznam zahtev izdelka, da bi ta pravilno odražal vse spremembe, ki nastajajo zaradi potreb končne stranke, zaradi novih idej, tehničnih izboljšav in drugih podobnih razlogov. Razvojna ekipa predstavniku naročnika podaja ocene potrebnega napora za posamezne predmete seznama, predstavnik naročnika pa postavlja ocene poslovnih vrednosti posameznih predmetov (vrednost predmeta za naročnika). Z uporabo teh dveh ocen (in mogoče še oceno tveganj) predstavnik naročnika nastavi prioritete predmetov v seznamu zahtev izdelka, tako, da na vrh postavi predmete z najvišjo oceno vrednosti, ki zahtevajo najmanj napora (in zmanjšajo tveganje). Tudi ocene se posodabljajo po vsakem sprintu upoštevajoč nova znanja in izkušnje, čemur se nato zopet prilagodi prioritetni vrstni red predmetov.

Scrum ne določa obliko ocen na seznamu zahtev izdelka, pogosta praksa je, da se uporabljajo relativne ocene v obliki točk namesto absolutnih vrednosti v obliki števila ur. Takšno točkovanje potrebnega napora za posamezne predmete nato čez čas omogoča, da se izmeri hitrost razvojne ekipe v obliki števila povprečno rešenih točk na posamezen sprint. S tem podatkom pa je nato tudi lažje napovedovati datum pričakovane izdaje končnega izdelka ali napovedati, koliko funkcionalnosti bo realiziranih do določenega datuma. Število dokončanih točk v enem sprintu označuje hitrost razvojne ekipe. Predmeti seznama zahtev se lahko precej razlikujejo v obsegu potrebnega napora. Večji predmeti se lahko zato razbijejo v več manjših in obratno (več manjših predmetov združimo v enega večjega) (Sutherland, 2010, str. 19).

Slika 3: Primer seznama zahtev izdelka

					Nove ocene napora			
					Preostali sprinti			
točka	Podrobno (url)	Prioritete	Ocena vrednosti	Začetna ocena napora	1	2	3	4
Kot kupec si želim dodati knjigo v nakupovalno košarico (glej skico ui na Wiki...)	...	1,0	7,0	5,0				
Kot kupec si želim odstraniti knjigo iz nakupovalne košarice	...	2,0	6,0	2,0				
Izboljšaj hitrost procesiranja transakcij	...	3,0	6,0	13,0				
Preuči rešitve za pohitritev validacije kreditnih kartic	...	4,0	5,0	20,0				
Posodobi vse strežnike na Apache 2.2.3	...	5,0	2,0	13,0				

Vir: J. Sutherland, Scrum Handbook, 2010, str.18.

Kot že omenjeno, so predmeti na seznamu zahtev izdelka razporejeni po vrednosti, tveganju, prioriteti in nujnosti. Prvi (najvišji) predmeti na seznamu določajo takojšnje razvojne aktivnosti. Višji predmeti na seznamu so tudi bili bolj obravnavani in o njih in njihovih vrednostih obstaja večje soglasje. So bolj jasni in bolj podrobno opisani ter posledično natančneje ocenjeni (zaradi večje jasnosti predmeta in podrobnejših opisov). Tudi če na nekem izdelku dela več Scrum ekip hkrati, je seznam zahtev izdelka vedno samo eden, predmete seznama pa se razporedi glede na določene lastnosti. Vzdrževanje seznama zahtev izdelka je stalen proces, ki vključuje dodajanje podrobnosti, ocen in popravljanje vrstnega reda predmetom. Pri tem sodelujeta predstavnik naročnika in razvojna ekipa. Občasne

aktivnosti, namenjene vzdrževanju seznama zahtev izdelka med sprintom naj ne porabijo več kot 10% zmogljivosti razvojne ekipe. Predstavnik naročnika, ki je odgovoren za seznam zahtev izdelka, lahko vedno posodobi predmete na seznamu (Schwaber & Sutherland, 2011, str. 12).

Za opis zahtev na seznamu zahtev izdelka se pogosto uporablja oblika »uporabniških zgodb« (izvirno »User Stories«), v bolj zahtevnih okoljih (npr.: za življenjsko kritično programsko opremo za »U.S. Food and Drug Administration«) pa se zahteve opiše z metodo »primeri uporabe« (izvirno »Use Case«) (Sutherland, 2010, str. 19).

Glavna korist uporabniških zgodbe je njihova jedrnatost in primernost, da se zahteva opiše v obliki dovolj kratkega teksta, da je ta lahko zapisan na manjšem kosu kartona (angl.: *Index Card*) ali papirja (angl.: *Sticky Notes*). Uporabniška zgodba je opis zahteve iz perspektive uporabnika in njegove interakcije s programsko opremo. Zato je tudi priporočljivo, da pri pisanju teh zahtev aktivno sodelujejo tudi končni uporabniki oz. naročnik (Pries & Quigley, 2011, str. 19).

Točke uporabniških zgodb (izvirno »Story Points«) so način za podajanje ocen obsega dela, ki ga je potrebno opraviti. Vsako uporabniško zgodbo na seznamu zahtev izdelka se opremi s številom točk. Točka predstavlja enoto merjenja, ki si jo samovoljno definira razvojna ekipa. Točka uporabniške zgodbe lahko predstavlja več dni dizajniranja, kodiranja in testiranja, odvisno od odločitve razvojne ekipe. Ekipa se lahko tudi odloči, da točka predstavlja zgolj dneve ali ure dela. Razvojna ekipa oceni, koliko točk uporabniških zgodb lahko doseže v sprintu, to pa potem določa, koliko uporabniških zgodb, ki vsebujejo te točke, se lahko realizira v enem sprintu (Pries & Quigley, 2011, str. 27).

2.3.2 Seznam zahtev sprinta

Seznam zahtev sprinta definira delo oz. naloge, ki jih izbere razvojna ekipa iz seznama zahtev izdelka in bodo izdelane v naslednjem sprintu ter bodo prinesle novo posodobitev končne programske rešitve oz. izdelka. Začetni seznam teh nalog razvojna ekipa pripravi v drugem delu sestanka za načrtovanje sprinta. Delo mora biti razdeljeno med naloge, ki naj bodo dolge od 4 do največ 16 ur. Daljše naloge naj se štejejo zgolj kot ogrodje nalog, ki še niso bile primerno definirane in razdrobljene. Samo razvojna ekipa lahko spreminja seznam zahtev sprinta. Seznam zahtev sprinta je v realnem času vidna slika dela, ki ga razvojna ekipa namerava doseči skozi sprint (Schwaber, 2004, str. 24).

S seznamom zahtev sprinta razvojna ekipa napove funkcionalnosti, ki bodo izdelane v naslednji razširitvi izdelka in napove delo, ki je potrebno za dokončanje teh funkcionalnosti. V seznamu je vidno celotno delo, ki je po mnenju razvojne ekipe potrebno za dosego cilja sprinta, in tako vsebuje načrt ekipe za uresničitev cilja. Načrt vsebuje dovolj podrobnosti, da so spremembe v napredku razumljene vsem na dnevnem Scrum-u. Razvojna ekipa seznam zahtev sprinta spreminja skozi celotni sprint. Spreminja ga, kadar načrtuje in spoznava več o delu, ki je potrebno za dosego cilja sprinta.

Vsako novo zahtevano delo se doda na seznam zahtev sprinta. Ko se delo izvaja in so naloge končane, se posodablja ocena preostalega dela. Sproti se odstranjuje elemente načrta za katere se ugotovi, da so nepotrebni (Schwaber & Sutherland, 2011, str. 13).

Slika 4: Primer seznama zahtev sprinta

				Nove ocene napora			
				Preostali sprinti			
Točka seznama zahtev izdelka	**Naloga sprinta**	**Prostovoljec**	**Začetna ocena napora**	**1**	**2**	**3**	**4**
Kot kupec si želim dodati knjigo v nakupovalno košarico (glej skico ui na Wiki...)	popravi bazo		5,0				
	kreiraj spletno stran (UI)		8,0				
	kreiraj spletno stran (JavaScript logika)		13,0				
	napiši avtomatski test sprejemljivosti		13,0				
	posodobi stran s pomočjo za kupce		3,0				
	...						
Izboljšaj hitrost procesiranja transakcij	zlij DCP kodo in končaj »layer-level« test		5,0				
	končaj strojno zaporedje za pRank		8,0				
	popravi DCP in uporabi pRank http API		13,0				

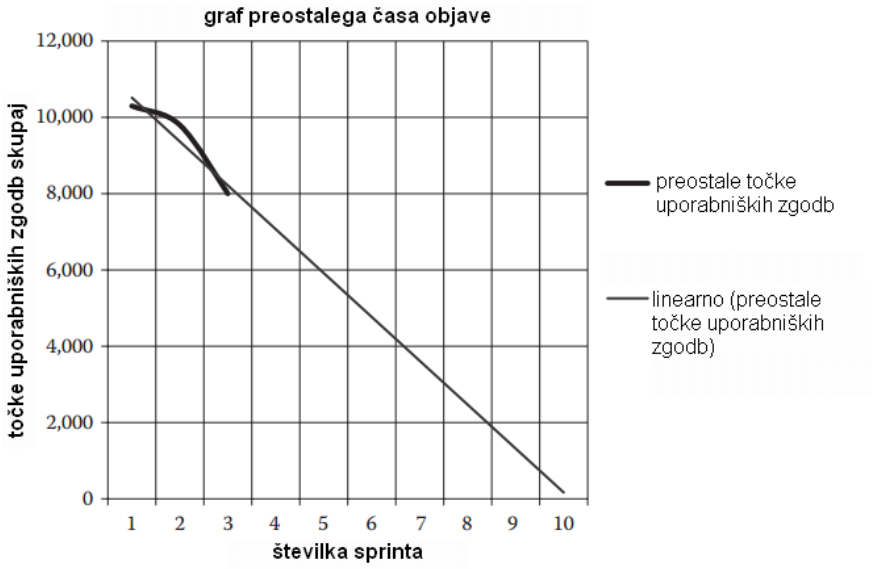
Vir: J. Sutherland, Scrum Handbook, 2010, str.22.

2.3.3 Graf preostalega časa (Burndown Chart)

Graf preostalega časa je grafična predstavitev dokončanja nalog. Tipični graf preostalega časa prikazuje potrebne ure za naloge, ki jih je še potrebno izpolniti proti dejanskim uram, ki so planirane za vse naloge. Graf nam omogoča, da zelo enostavno vidimo, če delo še poteka po planu (Pries & Quigley, 2011, str. 38).

Poznamo več primerov uporabe grafa preostalega časa. Če želimo z grafom prikazati količino preostalega dela ob začetku vsakega sprinta do dokončanja vseh zahtev na seznamu zahtev izdelka, uporabimo graf preostalega časa objave (angl.: *Release Burndown Chart*). V grafu izrišemo vsoto vseh točk uporabniških zgodb (angl.: *Story Points*) vseh nedokončanih uporabniških zgodb iz seznama zahtev izdelka. Prikaže nam korelacijo med obsegom preostalega dela in napredkom ekipe pri zmanjševanju tega preostalega dela (Mahnič & Žabkar, 2012, str. 75).

Slika 5: Primer grafa preostalega časa objave

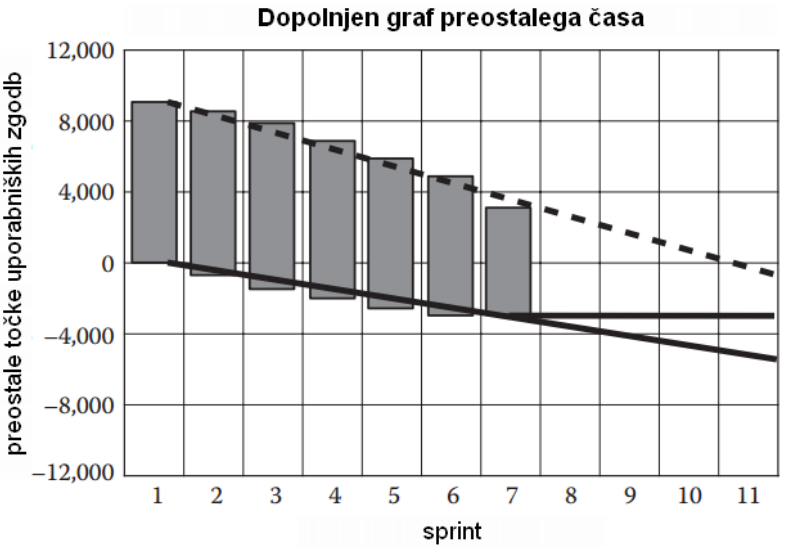


Vir: J. Schiel, Enterprise-Scale Agile Software Development, 2010, str. 235.

Na sliki 5 vidimo primer grafa preostalega časa objave, kjer je predvideno dokončanje 10.300 točk uporabniških zgodb v desetem sprintu. Predvidevanje je postavljeno na podlagi izmerjenih 1.150 točk povprečno opravljenih v enem sprintu (Schiel, 2010, str. 235). Svetlejša ravna črta predstavlja idealno linijo dokončanih točk (delo bi potekalo po planu). Temnejša črta pa prikazuje dejansko število dokončanih točk in nakazuje stanje projekta.

Ker se zahteve na seznamu zahtev izdelka lahko spreminjajo skozi projekt (se dodajajo ali odvzemajo) se primerno tudi spreminja idealna linija na grafu. Da bi lahko grafično spremljali tudi te spremembe, se graf preostalega časa objave dopolni, da zgleda tako, kot primer na naslednji sliki.

Slika 6: Primer dopolnjenega grafa preostalega časa objave

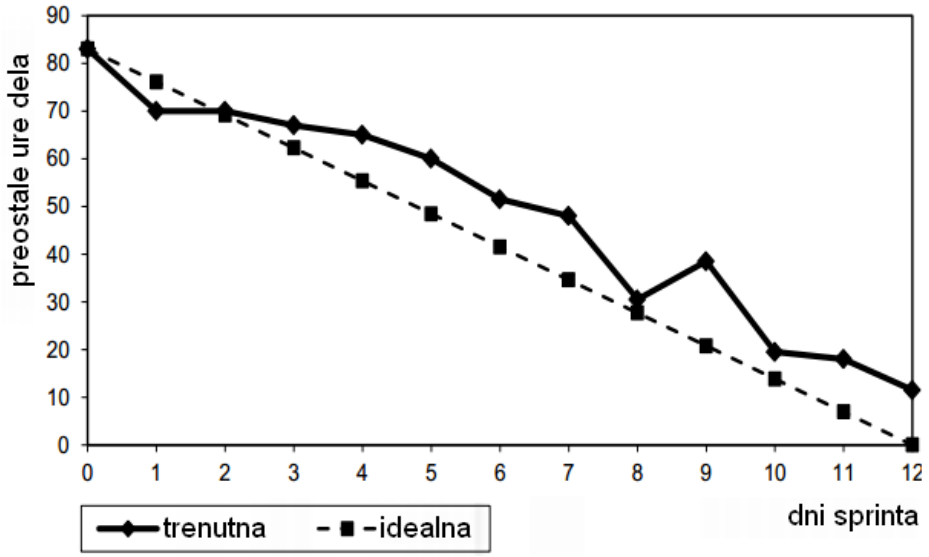


Vir: J. Schiel, Enterprise-Scale Agile Software Development, 2010, str. 237.

V tem grafu je velikost prvotnega seznama zahtev izdelka vidna kot dolžina sivega kvadrata nad izhodiščno X osjo (0), vse dodane zahteve, ki povečajo velikost seznama zahtev, so vidne, kot podaljšanje sivega kvadrata pod izhodiščno os X. Na začetku prvega sprinta so vse točke uporabniških zgodb štejejo kot prvotne, in se zato celoten siv kvadrat nahaja nad osjo X in šteje 10.300 točk. Ko projekt prispe do šestega sprinta, ekipi uspe število točk uporabniških zgodb zmanjšati na približno 6.000, a dolžina sivega kvadrata v sprintu 6 je še vedno vredna več kot 7.000 točk. Vsaj 3.000 točk tega kvadrata leži pod izhodiščno osjo X in več kot 4.000 nad osjo X. Na seznam zahtev izdelka je torej bilo dodanih teh 3.000 točk, ki se nahajajo pod osjo X. Črtkana linije trenda nam v tem grafu tako nakazuje trend zmanjševanja točk uporabniških zgodb (neodvisno ali so bile odstranjene zaradi odvzetih zahtev iz seznama zahtev izdelka, ali pa jih je razvojna ekipa dejansko dokončala). Polna linija pa prikazuje trend dodajanja novih zahtev v seznam zahtev izdelka (Schiel, 2010, str. 237).

Drugi primer uporabe grafa preostalega časa je graf preostalega časa sprinta. Gre za podoben graf kot graf preostalega časa objave, le da nam graf preostalega časa sprinta prikazuje količino preostalega dela, ki ga je potrebno dokončati do konca sprinta. Horizontalna os prikazuje število dni sprinta, na vertikalni osi pa izrisujemo število ur preostalega dela. Graf lahko dnevno posodabljamo tako, da zbiramo ocene števila ur, potrebnih za opravljanje dela na preostalih nedokončanih nalogah seznama zahtev sprinta. Ocene se zbirajo oz. beležijo na dnevnem Scrum sestanku. Linija trenda na grafu nakazuje, ali bo razvojna ekipa uspela do konca sprinta doseči dokončanje vseh zadanih nalog (Mahnič & Žabkar, 2012, str. 75).

Slika 7: Primer grafa preostalega časa sprinta



Vir: V. Mahnič & Žabkar, Measuring Progress of Scrum-based Software Projects, 2012, str. 75.

Na zadnji sliki je primer grafa preostalega časa sprinta. Črtkana črta prikazuje idealno linijo opravljenih ur, ki sledi planu sprinta. Polna črta prikazuje dejansko stanje opravljenih ur dela. V konkretnem primeru ekipa ob koncu sprinta ni uspela izvesti vseh planiranih ur. To se je nakazovalo že prej, saj je bila črta dejanskega stanja večinoma nad črto idealne linije.

2.3.4 Definicija »dokončanega«

Ko predmet s seznama zahtev izdelka označimo kot »dokončanega«, mora biti vsem članom ekipe enako razumljivo, kaj »dokončano« pomeni. Znotraj Scrum ekipe mora biti doseženo skupno razumevanje »dokončanega«, ki je za vsak projekt lahko drugačno. Takšen konsenz zagotavlja boljšo preglednost, poimenujemo ga »definicija dokončanega« in se uporablja za ugotavljanje, kdaj je delo ali objava izdelka dejansko dokončana (izpolnjuje vse definirane pogoje »dokončanega«).

Z definicijo »dokončanega« si razvojna ekipa tudi pomaga, ko določa, koliko predmetov s seznama zahtev izdelka bo uvrstila na seznam zahtev sprinta za naslednji sprint. Ob vsakem sprintu se ustvari razširitev funkcionalnosti končne programske opreme, ki mora biti pripravljena za potencialno izdajo in tudi v skladu s pripadajočo definicijo »dokončanega«. Predstavnik naročnika odloča, ali bo izdelana razširitev ob koncu sprinta tudi objavljena, vendar ne glede na njegovo odločitev mora biti razširitev skladna z definicijo »dokončanega« (temeljito testirana, delujoča, dokumentirana ipd.).

Scrum ekipa z vsakim projektom in tudi sprintom raste in se izboljšuje. Skladno s tem se nadgrajuje tudi definicija »dokončanega«, ki s časom vključuje strožja merila in tako izboljšuje kakovost (Schwaber & Sutherland, 2011, str. 15).

2.4 Scrum pravila ali »pravila igre«

Ena od odgovornosti skrbnika metodologije je tudi, da poskrbi, da vsakdo, ki je kakorkoli povezan s projektom (»piščanec« ali »prašič«) sledi Scrum pravilom. Ta pravila držijo Scrum proces skupaj tako, da je vsakdo seznanjen, kako delovati znotraj projekta. Če pravila niso dovolj dobro uveljavljenja, se izgublja čas z ugotavljanjem kaj storiti, po drugi strani pa izpodbijanje pravil pomeni izgubljanje časa s čakanjem na rešitve. Pravila, ki jih uvaja Scrum, so delovala že na tisočih uspešnih projektih. Vsaka želja po spremembi pravil je tema za diskusijo na sestanku za retrospektivo sprinta. Spremembe pravil naj izvirajo iz ekipe in ne iz vodstva oz. menedžmenta. Spremembe se lahko uveljavijo samo, če je skrbnik metodologije dovolj prepričan, da vsi vključeni dovolj dobro razumejo, kako deluje Scrum in so toliko izurjeni, da lahko spreminjajo pravila. Nobeno pravilo se ne more spremeniti, dokler skrbnik metodologije ne verjame, da smo dosegli takšen nivoja razumevanja Scrum-a (Schwaber, 2004, str. 121).

2.4.1 Sestanek za načrtovanje sprinta

Sestanek za načrtovanje sprinta je časovno omejen na 8 ur in sestavljen iz dveh delov (vsak po 4 ure). Prvi del je namenjen izbiranju predmetov iz seznama zahtev izdelka, drugi del pa za pripravo seznama zahtev sprinta (Schwaber, 2004, str. 121):

- Udeleženci so: skrbnik metodologije, predstavnik naročnika in razvojna ekipa. Zunanje osebe so lahko povabljene (s strani kateregakoli od prej naštetih) za zagotavljanje ali razlago informacij in nasvetov vsebinske ali tehnološke domene. Ko podajo svoje

razlage, zapustijo sestanek. Na sestanku niso prisotne osebe iz vlog »piščancev« (niti kot opazovalci).
- Seznam zahtev izdelka mora predstavnik naročnika pripraviti pred sestankom. V primeru odsotnosti predstavnika naročnika (in seznama zahtev izdelka) mora skrbnik metodologije zagotoviti primeren seznam zahtev izdelka in nadomestiti predstavnika naročnika.
- Cilj prvega dela sestanka (prvih štirih ur) je, da razvojna ekipa izbere tiste predmete seznama zahtev izdelka, ki verjame, da jih lahko pretvori v funkcionalnosti potencialno nove izdaje programske opreme. Te funkcionalnosti bo razvojna ekipa demonstrirala predstavniku naročnika in drugim interesnim skupinam na sestanku za revizijo sprinta ob koncu sprinta.
- Razvojna ekipa lahko podaja svoje predloge. Kaj s seznama zahtev izdelka pa bo na koncu dejansko predstavljajo sprint, pa je odgovornost predstavnika naročnika.
- Razvojna ekipa je odgovorna, da določi kolikšen del seznama zahtev izdelka, ki ga predstavnika naročnika določi za obdelavo, bo poskušala izdelati tekom sprinta.
- Časovni okvir štirih ur pomeni, da je to ves čas, ki je na voljo za analiziranje seznama zahtev izdelka. Nadaljnje analize naj se izvajajo med sprintom.
- Drugi del sestanka nastopi takoj po prvem delu in je prav tako omejen na 4 urni časovni okvir.
- Predstavnik naročnika mora biti razvojni ekipi na voljo (dosegljiv) za podajanje odgovorov glede mogočih nejasnosti na seznamu zahtev izdelka.
- Razvojna ekipa mora samostojno, brez katerihkoli direkcij od zunaj, odločiti, kako bo izbrane predmete seznama zahtev izdelka pretvorila v nove funkcionalnosti programske opreme zrele za potencialno objavo. Nihče drug ne sme sodelovati, dovoljeno je samo opazovanje in odgovarjanje na vprašanja.
- Rezultat drugega dela sestanka je seznam imenovan seznam zahtev sprinta. To je seznam aktivnosti, ocen in nalog, ki bo postavil razvojno ekipo na delovni tir. Seznam aktivnosti mogoče še ne bo izpopolnjen, vendar bo dovolj definiran za začetno delo na sprintu, med katerim bo razvojna ekipa oblikovala še ostale naloge seznama zahtev sprinta.

2.4.2 Dnevni scrum

Dnevni Scrum sestanek je časovno omejen na 15 minut ne glede na število članov razvojne ekipe (Schwaber, 2004, str. 123):

- Dnevni Scrum sestanek naj se vsak dan zgodi ob isti uri in na istem prostoru. Najbolje je, da je to prva stvar vsakega delovnega dne, tako da člani razvojne ekipe ob pričetku dela najprej razmislijo, kaj so počeli prejšnji dan in kaj bodo počeli danes.
- Dnevnega Scrum sestanka se morajo udeležiti vsi člani razvojne ekipe. Če se kdo ne more udeležiti sestanka, naj prisostvuje preko telefona (ali drugega komunikacijskega kanala), ali pa naj v njegovem imenu nekdo drug poroča o opravljenem in pričakovanem delu.

- Vsi morajo prispeti točno. Skrbnik metodologije prične s sestankom tudi, če še vsi niso zbrani. Član ekipe, ki zamudi, mora ob prihodu takoj plačati globo za zamudo v višini enega evra.
- Skrbnik metodologije prične sestanek s članom, ki je prvi na njegovi levi. Nato pa nadaljuje v redu urinega kazalca, dokler vsi člani ekipe ne pridejo na vrsto.
- Vsak član ekipe naj samo kratko odgovori na naslednja tri preprosta vprašanja:
    - Kaj je bilo zadnji dan storjenega s tvoje strani na projektu?
    - Kaj boš počel na projektu do naslednjega dnevnega Scrum sestanka?
    - Kaj ti preprečuje, da bi svoje delo opravil na najbolj učinkovit način?
- Člani ekipe naj se pri odgovarjanju na ta tri vprašanja ne spuščajo v probleme, dizajn, diskusije o problemih in podobne razprave. Skrbnik metodologije je zadolžen, da sestanek poteka hitro in poročanje teče tekoče od člana do člana.
- Med sestankom hkrati poroča samo ena oseba. Ostali samo poslušajo brez stranskih pogovorov.
- Če član ekipe med sestankom poroča o nečem, kar je interesantno tudi drugim članom, ali pa rabi njihovo pomoč, lahko katerikoli član ekipe skliče poseben sestanek takoj po dnevnem Scrum-u in na ta sestanek povabi vse zainteresirane.
- »Piščancem« ni dovoljeno, da se oglašajo na sestanku ali so na kakršenkoli drug način vsiljivi.
- Osebe z vlogo »piščancev« naj med sestankom stojijo nekje ob strani, da ne morejo motiti poročanja.
- Če je zunanjih opazovalcev (»piščancev«) preveč, lahko skrbnik metodologije omeji število teh oseb in tako ohrani potreben red in osredotočenost na potek sestanka.
- Prav tako je osebam zunaj ekipe prepovedano, da po sestanku članom ekipe sugerirajo kakšne razlage, navodila in podobno.
- Osebe, ki se ne morejo držati zgornjih pravil, se lahko izključi s sestanka ali iz ekipe.

2.4.3 Sprint

Sprint je časovno omejen na največ 30 zaporednih dni. To je količina časa v katerem razvojna ekipa zgradi nekaj, kar je v interesu predstavnika naročnika in drugih interesnih skupin in pripelje izdelek v stanje, ko je potencialno pripravljen za novo izdajo. To je tudi največji obseg časa, v katerem razvojna ekipa še opravlja delo brez potrebe po posebnih artefaktih ali dokumentaciji za podporo. Je tudi največji časovni obseg, ki še zagotavlja, da interesne skupine ne izgubijo interesa v delo razvojne skupine, in da še ohranjajo zaupanje, da razvojna ekipa razvija izdelek v njihovo korist (Schwaber, 2004, str. 125):

- Razvojna ekipa lahko poišče tudi zunanjo pomoč, informacije ali podporo med izvajanjem sprinta.
- Nikomur od zunaj ni dovoljeno, da razvojno ekipo kakorkoli usmerja, ji daje napotke ali komentarje o njenem delu. Razvojna ekipa je popolnoma samo-organizirana.
- Razvojna ekipa je zavezana seznamu zahtev izdelka med časom sprinta. Med sprintom nihče ne sme spreminjati seznama zahtev izdelka, ta je do konca sprinta zamrznjen.

- Če se sprint izkaže, da ni več veljaven, smiseln ali uporaben, ga lahko izjemoma skrbnik metodologije tudi prekine in vzpostavi nov sestanek za načrtovanje sprinta. Ta poseg lahko skrbnik metodologije sproži na lastno zahtevo ali na zahtevo predstavnika naročnika ali razvojne ekipe. Tak poseg je smiseln, če se izbrana tehnologija izkaže kot neuporabna, če se spremenijo vsebinski pogoji toliko, da sprint nebi bil več v korist organizaciji, ali če je razvojna ekipa preveč motena od zunanjih oseb med sprintom.
- Če razvojna ekipa ugotovi, da ne bo mogla izpolniti vseh zavez s seznama zahtev izdelka, lahko predstavnika naročnika zaprosi, da pove, katere predmete bi se dalo izključiti iz trenutnega sprinta. Če je teh predmetov preveč, lahko sprint postane nesmiseln in skrbnik metodologije ga lahko izjemoma prekine.
- Če razvojna ekipa ugotovi, da bo med sprintom zmožna izdelati več funkcionalnosti, kot je bilo predvideno ob začetku sprinta, se lahko s predstavnikom naročnika dogovori o dodatnih predmetih s seznama zahtev izdelka, ki se naj vključijo v tekoči sprint.
- Razvojna ekipa ima med sprintom dve administrativni zahtevi: izvajanje dnevnega Scrum sestanka in upravljanje s seznamom zahtev izdelka tako, da se ta redno posodablja in je objavljen tako, da je viden vsem zainteresiranim v projekt. Nove zasnovane naloge se naj dodajo na seznam zahtev sprinta. Ocene potrebnih ur na nalogah pa naj bodo dnevno osvežene.

2.4.4 Revizija sprinta

Sestanek za revizijo sprinta je časovno omejen na štiri ure (Schwaber, 2004, str. 126):

- Ekipa ne sme porabiti več kot uro časa za pripravo na sestanek za revizijo sprinta.
- Namen revizije sprinta je, da razvojna ekipa predstavniku naročnika in drugim interesnim skupinam predstavi funkcionalnosti, ki so bile dokončane med sprintom in ustrezajo definiciji »dokončanega«. Definicija »dokončanega« mora pri tem biti razumljiva tudi predstavniku naročnika in zainteresiranim.
- Funkcionalnosti, ki niso dokončane (po definiciji »dokončanega«), ne morejo biti predstavljene.
- Artefakti, ki še niso funkcionalni, naj se ne predstavljajo, razen če je to potrebno za predstavitev drugih dokončanih funkcionalnosti. Takšni artefakti tudi ne morejo biti predstavljeni kot nekaj, kar je še v procesu izdelave (angl.: *Work in Progress*), da pri predstavitvi ne zmedejo zainteresiranih in sprožijo potrebe po dodatnih razlagah.
- Revizija sprinta se prične s članom razvojne ekipe, ki najprej predstavi cilj sprinta in zahteve s seznama zahtev izdelka, za katere so se zavezali v okviru tega sprinta in ki so jih dokončali. Različni člani razvojne ekipe lahko diskutirajo, kaj je šlo med sprintom dobro in kje so bile težave.
- Glavnina sestanka se porabi za predstavitev funkcionalnosti in odgovarjanje na vprašanja, ki se nanašajo na to predstavitev.
- Zainteresirane se ob koncu povabi, da podajo svoja mnenja, občutke in želje po morebitnih spremembah in njihovih prioritetah.
- Glede na odzive, se predstavnik naročnika s člani razvojne ekipe in zainteresiranimi dogovarja o morebitnih prerazporeditvah zahtev na seznamu zahtev izdelka.

- Med predstavitvijo lahko interesne skupine podajajo komentarje, opazke ali kritike glede novih funkcionalnosti programske opreme.
- Interesenti lahko opazijo, da nekatere funkcionalnosti, ki so bile pričakovane, manjkajo in zahtevajo, da se te funkcionalnosti uvrstijo na seznam zahtev izdelka prioritetno.
- Prav tako se lahko podajo zahteve za uvrstitev na seznam zahtev izdelka funkcionalnosti, ki se interesentom med predstavitvijo zazdijo potrebne.
- Skrbnik metodologije mora ugotoviti število zainteresiranih za udeležbo na sestanku revizije sprinta in doseči, da je prisotnost vsem omogočena.
- Na koncu sestanka skrbnik metodologije napove naslednji sestanek revizije sprinta in določi datum ter prostor.

2.4.5 Retrospektiva sprinta

Sestanek za retrospektivo sprinta je časovno omejen na tri ure (Schwaber, 2004, str. 128):

- Udeležijo se ga lahko samo razvojna ekipa, skrbnik metodologije in predstavnik naročnika. Prisotnost predstavnika naročnika ni nujna.
- Sestanek se prične z razvojno ekipo, pri tem pa vsi njeni člani odgovorijo na dve vprašanji:
    - Kaj je med sprintom potekalo dobro?
    - Kaj lahko še izboljšamo v naslednjem sprintu?
- Skrbnik metodologije odgovore članov razvojne ekipe zapiše v obliki povzetka.
- Razvojna ekipa napove, v kakšnem vrstnem redu želijo diskutirati o potencialnih izboljšavah.
- Skrbnik metodologije na sestanku ni prisoten zato, da bi odgovarjal na vprašanja, ampak zato, da svetuje in olajša razvojni ekipi, da ta sama poišče načine, kako bi Scrum proces za njih deloval najbolje.
- Koristni predlogi, ki se lahko izvajajo v naslednjemu sprintu, se oblikujejo kot visoko prioritetni nefunkcionalni predmeti seznama zahtev izdelka. Retrospektive, ki ne prinašajo sprememb, so nepredmetne in frustrirajoče.

2.5 Planiranje

Ocenjevanje napora, ki je del aktivnosti planiranja, je vedno neka vrsta ugibanja, ki lahko močno variira v kakovosti. Včasih se premalo razmišlja o aktivnostih, ki so potrebne za produkcijo rezultatov. To omejitev lahko premagamo s premišljenim seciranjem rezultatov v sestavne naloge. Pogoste napake pri planiranju so poskusi planiranja za predolga obdobja v prihodnost. Problem lahko zmanjšamo s spremljanjem odstopanj v ocenah med izvajanjem projekta (Pries & Quigley, 2011, str. 22). Ocenjevanje predmetov seznama zahtev izdelka nam omogoča, da razumemo njihovo grobo velikost in potreben napor za njihovo izdelavo. To je uporabno iz dveh razlogov: olajšuje nastavljanje prioritet in omogoča napovedovanje in spremljanje napredka projekta.

V Scrum metodologiji razlikujemo dvoje ocenjevanj: grobo ocenjevanje na seznamu zahtev izdelka in bolj natančno ocenjevanje za seznam zahtev sprinta. Predmeti na seznamu zahtev izdelka se ocenijo, ko dodajamo nove predmete, ko spreminjamo obstoječe, ali ko se razumevanje velikosti predmeta spremeni (Pichler, 2010, str. 64).

Poznamo več pristopov za planiranje in ocenjevanje napora potrebnega za dokončanje neke naloge. Nekaj teh pristopov bom opisal v nadaljevanju.

2.5.1 Zgodovinski podatki

Z uporabo zgodovinskih metod organizacija dejansko koristi svoje izkušnje na projektih, posameznikih in v ekipah. Negotovost je mogoče zmanjšati, če poznamo zmogljivosti iz preteklosti (predvidimo, da zmogljivost iz preteklosti napoveduje zmogljivosti v prihodnosti). Uporaba zgodovinskih podatkov ima omejitve, kadar se srečujemo s popolnoma novimi aktivnostmi, ki jih še prej nikoli nismo izvajali, ali pa se ukvarjamo z novo tehnologijo ali novim okoljem. V vsakem primeru pa je mogoče zmanjšati tveganje, če novo situacijo primerjamo z vidikom iz preteklosti (Pries & Quigley, 2011, str. 25).

2.5.2 Uporaba ekspertov

Če v organizaciji najdemo ljudi, ki imajo izkušnje s področja predmeta, ki ga ocenjujemo, jih lahko uporabimo za podajo ocen. Če je na voljo več oseb z iskanimi izkušnjami, lahko uporabimo kakšno metodo ocenjevanja, kot na primer »Planning Poker«. Razlike med podanimi ocenami strokovnjakov oz. ekspertov nam dajo dober pogled na negotovost ocen, urnikov in proračuna (Pries & Quigley, 2011, str. 26).

2.5.3 Točke uporabniških zgodb (angl.: *Story Points*)

Točke uporabniških zgodb bazirajo na kratkih opisih kompleta funkcionalnosti imenovanega uporabniška zgodba (angl.: *User Story*). Uporabniške zgodbe imajo naslednje lastnosti (Pries & Quigley, 2011, str. 27):

- opis funkcionalnosti s strani člana ekipe ali s strani stranke oz. uporabnika,
- uporablja se predloga za izboljšanje konsistence v komunikaciji,
- vsebujejo predmete, ki se jih da videti in preizkusiti na sestanku za revizijo sprinta,
- so ocenjene z urami, dnevi ali točkami uporabniških zgodb,
- so neodvisne od drugih uporabniških zgodb.

Točke uporabniških zgodb so grobo definirana relativna merila surovega napora. Predmet vreden eno uporabniško točko je za polovico manjši od predmeta, ki je vreden dve uporabniški točki. Predmet, ki je velik tri točke uporabniških zgodb zahteva toliko napora, kot predmet ene točke in predmet dveh točk skupaj. Relativna merila izkoriščajo prednost dejstva, da je tudi velikost sama relativna. Pogosto uporabljeni razredi velikosti uporabniških zgodb izraženi s številom točk so vidni v naslednji tabeli (Pichler, 2010, str. 65):

Tabela 1: Pogosto uporabljeni razredi velikosti uporabniških zgodb

Točke uporabniških zgodb	Oznaka razreda	
0,0	Predmet je že bil implementiran	
1,0	XS	»Extra Small«
2,0	S	»Small«
3,0	M	»Medium«
5,0	L	»Large«
8,0	XL	»Extra Large«
13,0	XXL	»Double Extra Large«
20,0	XXL	»Huge«

Vir: R. Pichler, Agile Product Management with Scrum, 2010 str. 65.

Nelinearno zaporedje v tabeli omogoča hitrejše odločanje v razvojni ekipi in preprečuje predolgo diskutiranje o pravi vrednosti neke ocene. Zaporedje se lahko tudi podaljša na večje razrede oz. si jih ekipa oblikuje čisto po svoje. Ekipa si naj izbere razrede, ki ji najbolj ustrezajo. Izbranih razredov se naj nato drži skozi celoten projekt. Ker so točke uporabniških zgodb relativne in jih ekipa določi samovoljno, jih ni mogoče uporabiti hkrati za druge ekipe ali projekte oz. niso primerljive s točkami, ki si jih izberejo druge razvojne ekipe (Pichler, 2010, str. 65).

Število točk uporabniških zgodb, ki je pripisano posamezni uporabniški zgodbi, predstavlja njeno splošno velikost. Določena formula za definiranje velikosti uporabniške zgodbe ne obstaja. Namesto formule je točka uporabniških zgodb združena ocena, ki vključuje obseg potrebnega napora za izdelavo funkcionalnosti, kompleksnost izdelave, povezana tveganja in druge podobne okoliščine.

Obstajata dva pogostejša načina, kako se lotiti točkovanja. Prvi pristop priporoča izbiro uporabniške zgodbe, ki je najmanjša od vseh in zahteva najmanj napora. Takšno uporabniško zgodbo ocenimo z eno točko. Vse ostale pa ocenimo upoštevajoč to merilo (če je uporabniška zgodba enkrat bolj zahtevna od te najmanjše, dobi oceno dveh točk). Drugi pristop priporoča, da se izbere uporabniško zgodbo, ki se zdi nekako srednje velikosti, in se ji dodeli število točk, ki je v sredini območja točk, ki ga želimo uporabljati za ocenjevanje. Če želimo uporabiti območja točk od 1 do 10, potem tej izbrani uporabniški zgodbi dodelimo pet točk. Podobno kot pri prvem pristopu, potem vse ostale uporabniške zgodbe ocenimo s pomočjo primerjave s prvo ocenjeno uporabniško zgodbo. Najboljši način, da ugotovimo, kako takšno ocenjevanje deluje, je, da ga poizkusimo (Mahnič, 2011, str. 36).

2.5.4 Hitrost (angl.: *Velocity*)

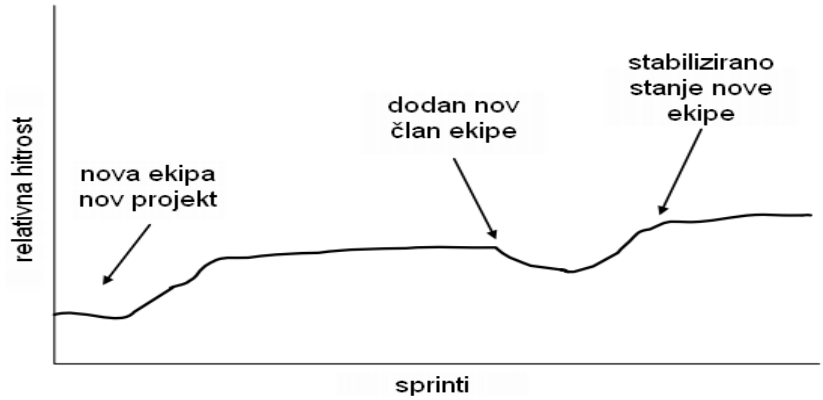
Hitrost ekipe je indikator količine dela, ki ga razvojna ekipa lahko opravi v enem sprintu. Omogoča nam, da spremljamo in napovedujemo napredek na projektu, in je zato uporaben pri planiranju. Bolj natančno hitrost opredelimo s seštevkom napora, ki so ga zahtevali rezultati dela, potrjenega s strani predstavnika naročnika, v enem sprintu.

Primer: na sestanku za načrtovanje sprinta se razvojna ekipa zaveže, da bo implementirala 6 uporabniških zgodb, ki skupaj štejejo 12 točk uporabniških zgodb napora. Ob koncu sprinta predstavnik naročnika natančno pregleda novo izdajo programske rešitve in ugotovi, da so bile izpolnjene vse zahteve glede na definicijo dokončanega, ki je sprejeta v ekipi. Razen pri eni uporabniški zgodbi (ocenjeni z dvema točkama) je manjkal večji del dokumentacije, ki je predviden v definiciji dokončanega. Ker ta uporabniška zgodba ni bila dokončana, njene ocenjene točke napora ne štejejo v izračun hitrosti. Seštevek točk uporabniških zgodb je tako 10. Hitrost razvojne ekipe za ta sprint pa je prav tako enaka 10.

Hitrost ekipe se najbolje določi z opazovanjem zmožnosti ekipe, da pretvarja predmete seznama zahtev izdelka v novo izdajo programske opreme. S takšnim merjenjem se uresničuje tudi načelo agilnosti iz manifesta agilnosti, ki pravi »*Primarno merilo napredka je delujoča programska oprema*«.

Hitrost razvojne ekipe se lahko spreminja, na kar vpliva več faktorjev (dinamika ekipe, ovire, razpoložljivost, drugi vplivi). Če nekaj članov ekipe odide na dopust ali zbolijo, bo hitrost ekipe padla. Zanimivo je opazovati, kako se hitrost ekipe spreminja, če dodajamo nove člane v razvojno ekipo. Ugotovitve kažejo, da se hitrost ekipe po vstopu novega člana stabilizira šele po dveh do treh sprintih (Pichler, 2010, str. 82).

Slika 8: Hitrost razvojne ekipe skozi čas



Vir: K. Pries & Quigley, Scrum Project Management, 2011, str. 34.

Hitrost razvojne ekipe je merilo za eno samo ekipo in ga ni mogoče primerjati s hitrostjo drugih ekip, saj ni zagotovila, da točke uporabniških zgodb med ekipama odražata enako količino napora. Točka uporabniških zgodb je namreč relativno merilo. Hitrosti dveh ekip bi bilo mogoče primerjati samo, če obe ekipi uporabljata poenotene točke uporabniških zgodb z enakim pomenom (Pichler, 2010, str. 83).

2.5.5 Poker planiranja (angl.: *Planning Poker*)

Poker planiranja je tehnika planiranja, ki omogoča učinkovito in ekipno ocenjevanje. Kombinira mnenja ekspertov, primerjave nalog in razčlembe nalog v prijeten pristop, ki prinaša hitre in zanesljive ocene. Pri pokru planiranja sodelujejo vsi člani razvojne ekipe (programerji, preizkuševalci, oblikovalci podatkovnih baz, analitiki, oblikovalci uporabniških

vmesnikov, …), kar pomeni, da najverjetneje ne bo vključenih več kot 10 oseb. Če vseeno presežemo to število, je bolje, da se ekipa razdeli na dve skupini. Predstavnik naročnika lahko sodeluje pri pokru planiranja, vendar ne podaja ocen.

Pred začetkom »igranja« je potrebno pripraviti karte. Za vsakega udeleženca se pripravi paket kart v katerem je na vsaki karti napisano pripadajoče število točk uporabniških zgodb (npr. karte s številkami: 0, 1, 2, 3, 5, 8, 13, 20, 40, in 100). Vsak udeleženec dobi torej v roke en tak paket kart. Za vsako uporabniško zgodbo oz. predmet s seznama zahtev moderator igre (navadno predstavnik naročnika ali katerakoli druga oseba) prebere njen opis in poda odgovore na vprašanja, ki se mogoče pojavijo na prebrani opis. Ko je opis predmeta vsem poznan in razumljen, se prične igra. Vsak udeleženec iz kompleta svojih kart izbere eno, ki predstavlja njegovo oceno za izbrani predmet. Karto samo izbere in jo še zadrži zase (jo ne pokaže ostalim igralcem). Ko vsi igralci izberejo svojo karto z oceno, vsi na enkrat izbrano karto razkrijejo (jo postavijo tako, da je vsem vidna).

Na tej točki je zelo verjetno, da se bodo točke na kartah med sabo precej razlikovale, kar je v bistvu dobro. Če se točke razlikujejo, morata igralca, ki sta podala največjo in najmanjšo oceno, njuno odločitev pojasniti. Pri tem je pomembno, da se samo poda pojasnitev. Ostali udeleženci naj razlago ne komentirajo preveč, jo ne izpodbijajo ali kako drugače napadajo podano oceno. Ostali udeleženci naj zgolj poslušajo in poskušajo razumeti, zakaj je igralec izbral tako visoko oz. nizko število točk. Celotna ekipa lahko nato še nekoliko diskutira o ocenjevanem predmetu, pri tem pa si lahko moderator zapisuje beležke, če meni, da je med diskusijo slišal kaj uporabnega za čas, ko se bo uporabniška zgodba dejansko implementirala.

Po diskusiji vsak igralec ponovno izbere eno izmed kart iz svojega kompleta. Enako kot v prvem krogu najprej vsi zase izberejo eno karto, nato pa vsi hkrati postavijo izbrane karte na vpogled. V veliko primerih bodo ocene na kartah v tem drugem krogu igranja že konvergirale v eno samo oceno oz. bodo točke na vseh prikazanih kartah že enake. Če se to še ne zgodi, se ponovi proces igranja (nov krog) in podajanja razlag igralcev, ki izberejo najvišjo ali najnižjo oceno. Cilj igre je, da priigramo situacijo, ko vsi udeleženci podajo enako število točk za oceno izbranega predmeta. Igro lahko pohitrimo tako, da ne odigramo ponovnega kroga, če se na primer v tretjem krogu zgodi, da odstopa samo karta enega igralca (ostale pa so enake). V takem primeru lahko moderator igralca, ki je izbral izstopajočo karto, vpraša, če se lahko sprejme ocena, ki jo je izbrala večina igralcev (Mahnič, 2011, str. 56).

Poker planiranja je mogoče igrati tudi samo z delom članov celotne razvojne ekipe. To sicer ni idealna situacija, vendar je včasih to bolj razumna odločitev, sploh v primerih, ko je potrebno oceniti večje število predmetov npr. ob začetku projekta. Takšno igranje najlažje izvedemo, če celotno ekipo razdelimo na dve ali tri manjše skupine, vsaka naj ima vsaj tri ocenjevalce. Pomembno je, da vse te skupine ocenjujejo dosledno. Tri točke v eni skupini morajo pomeniti enako kot tri točke v ostalih skupinah. Da to dosežemo, je dobro najprej opraviti nekaj iger skupaj s celotno ekipo in šele nato ekipo razdeliti na skupine za ocenjevanje ostalih predmetov. Kopije skupaj ocenjenih uporabniških zgodb naj ima vsaka skupina na vidnem mestu (Mahnič, 2011, str. 57).

Poker planiranja je dobra praksa ocenjevanja iz več razlogov. Kot prvo, ta metoda združi mnenja več ekspertov različnih disciplin na projektu, ki predstavljajo najprimernejše za podajo ocen. Nek predmet bodo najbolje ocenili tisti, ki bodo tudi zavezani, da ga na koncu tudi implementirajo. Kot drugo. Živahen dialog, ki se dogaja med igranjem pokra planiranja in ob razlagah najbolj izstopajočih ocenjevalcev, izboljšuje točnost ocen, še posebno pri predmetih z višjo negotovostjo. Ko igralce pozivamo, da razlagajo svoje ocene, s tem bolje odkrivamo skrite informacije, na katere ostali npr. niso pomislili. To je sploh uporabno za ocenjevanje uporabniških zgodb, ki so lahko nejasne. Kot tretje, so študije pokazale, da povprečenje posameznih ocen prinaša boljše rezultate, kakor tudi skupinsko odločanje o ocenah. Oba elementa najdemo v igranju poker planiranja. Nazadnje lahko poker planiranje označimo tudi kot zabaven pristop k drugače nepriljubljeni aktivnosti ocenjevanja nalog, kar mu vsekakor tudi lahko štejemo v prednost (Mahnič, 2011, str. 58).

Študija, ki sta jo opravila Mahnič in Hovelja (2012, str. 1) prav tako nakazuje, da poker planiranje prinaša bolj natančne ocene in tudi zmanjšuje preveč optimistično ocenjevanje, ki se zna pojaviti pri skupinskem ocenjevanju. Ugotovila sta, da se pojav preveč optimističnih ocen zmanjša, če imamo v ekipi igralcev pokra planiranja bolj izkušene ljudi oz. eksperte.

3 PROCES VODENJA PROJEKTOV V OPAZOVANEM PODJETJU

V opazovanem podjetju imamo proces vodenja projektov opisan v pravilniku o vodenju in izvajanju projektov (Pravilnik o vodenju in izvajanju projektov podjetja Comland d.o.o.). V pravilniku so tako zapisana vsa pravila, ki jih mora organizacijsko zadovoljiti projekt, ki se izvaja v organizaciji.

Pravilnik med drugim vsebuje predvsem:

- opise standardnih dogovorjenih pravil o načinu vodenja in izvajanja projektov,
- osnovno organiziranje projektov s standardnimi vlogami udeležencev na projektih,
- opredelitev standardov planiranja, spremljanja, vzdrževanja in poročanja na projektih,
- opredelitev o dogovorjenih enotnih postopkih in navodilih pri izvajanju projektov.

Pravilnik projekt opredeljuje kot enkraten ali trajen proces z opredeljenim namenom, ciljem in naročnikom, ter omejen s kadri, časom in stroški.

Razločuje med več tipi projektov:

- Plačljivi projekti:
    - enkratni razvojni projekt,
    - vzdrževalni projekt.
- Interni (stroškovni) projekti:
    - razvojni projekt,
    - prodajno – tržni projekt,
    - izobraževalni projekt,
    - fiktivni projekt.

Metodologija upravljanja projektov je v pravilniku zajeta kot opis funkcije vodenja projektov (projektni menedžment).

Funkcija vodenja projektov je razčlenjena z:

- deli projekta:
    - naloge,
    - aktivnosti,
    - rezultati,
    - viri.
- organizacijo oz. vlogami:
    - naročnik,
    - projektna skupina,
    - direktor projekta,
    - vodja projekta,
    - izvajalec,
    - zunanji izvajalec.
- življenjskim ciklom projekta:
    - priprava,
    - vzpostavitev,
    - izvedba,
    - zaključek projekta.
- dokumentacijo,
- orodji (informacijski sistem).

Vsa opravila, ki se izvajajo na projektu, se izvaja v sklopu nalog, ki jih praviloma izdaja vodja projekta ali pooblaščeni član projektne skupine. Z nalogo se določi izvajalce, odgovornosti, trajanje, pričakovani rezultat in stroške. Za lažje strukturiranje nalog na projektu se uporabljajo aktivnosti, ki združujejo skupke nalog v neko logično enoto. Pri manjših projektih je strukturiranje v aktivnosti nepotrebno oz. se aktivnost enači z nalogo. Z rezultati projekta se opiše predvidene izhode projekta. Rezultat projekta opisuje vsebino predvidenega izdelka, kriterij in rok dobave izdelka ter pogoje prevzema izdelka. Viri projekta opredeljujejo vire, ki se uporabijo za izvedbo projekta in pa finančna ter druga sredstva.

Organizacijska struktura projekta določa osnovne vloge, ki se pojavljajo na projektu:

Naročnik projekta je pri plačljivih projektih zunanja stranka, za katero se projekt izvaja, pri internih projektih pa naročnika projekta predstavlja projektni svet ali druga oseba iz menedžmenta, ki je pobudnik internega projekta. Od naročnika projekta se pričakuje jasna opredelitev zahtev, namena, ciljev in časovnih rokov projekta, ter odgovornost za prevzem izdelkov ob koncu projekta.

Projektna skupina je sestava ljudi oz. sodelavcev podjetja na projektu skupaj z zunanjimi sodelavci na projektu ter predstavniki naročnika.

Direktor podjetja je vloga, ki se vpelje predvsem pri večjih projektih sestavljenih iz pod-projektov in je zadolžena za koordinacijo vodij projektov na posameznih pod-projektih, ter za komunikacijo z zunanjimi interesnimi skupinami na nivoju krovnega projekta.

Operativno vodenje projekta opravlja **vodja projekta**. Med njegove naloge spada predvsem: opredeljevanje ciljev projekta, izvedba planov projekta, opredeljevanje organizacije projekta, skrb za dokumentacijo projekta, organizacijo aktivnosti in nalog na projektu, oblikovanje projektne skupine in njeno vodenje, komuniciranje (notranje in zunanje) in poročanje, skrb za pogodbo in izvajanje njenih določil, dobava rezultatov, upravljanje s spremembami, spremljanje ekonomičnosti na projektu in koordiniranje zunanjih pod-izvajalcev na projektu.

Izvajalce na projektu določa vodja projekta in jih tudi razporeja po nalogah projekta. Na nalogi so izvajalci odgovorni za izvedbo del, ki jih naloga predvideva. Ko posamezni izvajalec opravi njemu dodeljene naloge, ni več udeleženec projekta.

Pravilnik posebej opredeljuje tudi **zunanje izvajalce**. To so člani projektne skupine, ki ne prihajajo iz podjetja, ampak na projektih sodelujejo v kateri od možnih oblik zunanjega sodelovanja (podizvajalci, člani konzorcijev ali tehnološki partnerji).

Pravilnik loči štiri glavne faze v življenjskem ciklu projekta:

- **Faza I: priprava projekta:**
  S to fazo se opravi opredelitev projekta (namen, cilj, finančni projektni okvir, …) in se uredi formalna podlaga za izvedbo projekta. Končni rezultat te faze je interni vzpostavitveni dokument, ki mora biti tudi potrjen s strani projektnega sveta.
- **Faza II: vzpostavitev projekta:**
  V tej fazi vodja projekta definira projekt in uredi okolje za opredelitev, spremljanje in nadzor projekta. Obseg podrobnosti tega opredeljevanja je odvisno od zahtevnosti in kompleksnosti projekta.
- **Faza III: izvajanje projekta:**
  V tej fazi poteka dejansko izvajanje del na projektu po predvidenih planih in pogojih iz prejšnjih faz. Namen je izpolniti cilje, ohranjati namen, obdržati vpogled naročnika v projekt in nadzorovati spremembe. Predvidene so naslednje aktivnosti vodenja projekta:
    - delegiranje, prevzem, izvajanje in zaključevanje nalog;
    - usklajevanje operativnih planov izvedbe s temeljnimi plani projekta;
    - obveščanje (projektnega sveta, naročnika, …);
    - spremljanje porabljenih ur, virov in realizacije na projektu;
    - izpolnjevanje pogodbenih zahtev.
- **Faza VI: zaključek projekta:**
  Zaključek projekta nastopi, ko so izpolnjeni cilji in so dobavljeni in od naročnika potrjeni predvideni rezultati projekta. Projekt formalno zaključi projektni svet na podlagi zaključnega poročila, ki ga izdela vodja projekta.

  Za razvojne projekte so v pravilniku še podrobneje definirane posamezne standardne aktivnosti, ki se praviloma planirajo v fazi vzpostavitve projekta:

- **Razvoj:**
  - Analiza
  - Načrtovanje
  - Kodiranje
  - Testiranje
  - Dokumentiranje
- **Uvajanje:**
  - Namestitev
  - Usposabljanje, predstavitve, seminarji, delavnice
- **Vzdrževanje:**
  - Izdelava dopolnitev ali prilagoditev (če to ni izvedeno posebej v samostojnem projektu)
  - Odpravljanje napak (če to ni izvedeno posebej v samostojnem projektu)
  - Podpora uporabnikom (vsebinska in tehnična podpora, incidenti, servisi) (če to ni v posebnem projektu)

Posebej pravilnik obravnava tudi projekte jamčenja, v primerih, ko gre za samostojni projekt. Gre za projekte nudenja vzdrževanja za izdelano programsko opremo za določeno obdobje pod pogoji, določenimi v pogodbi.

Pravilnik opredeli dva tipa poročanja:

- Operativno poročanje:
  gre za poročanje znotraj projektne skupine posameznega projekta. Projektni vodja mesečno poroča nadrejenemu vodji na dogovorjen način.
- Četrtletno poročanje:
  to poročanje predvideva izdelavo četrtletnega poročila za predstavitev na četrtletnem kolegiju. Poročilo pripravi vodja projekta in vsebuje poročilo o finančnih rezultatih preteklega četrtletnega obdobja, prodajni vidik, vidik zadovoljstva naročnika o napredku projekta in interni vidik projekta (preteklo delo in predlogi za izboljšave).

V poglavju o dokumentaciji na projektih pravilnik opredeljuje zgolj standarde za formulacijo oz. oblikovanje dokumentov, identifikacijske standarde in vrste projektne dokumentacije (navodila in obrazci; strokovna, tehnična, funkcionalna in ostala projektna gradiva; poročila in zapisniki; pogodbe naročila in obračuni; ostalo). Pravilnik tudi določa, kam se mora dokumentacija shranjevati in predlaga osnovno strukturo hrambe projektne dokumentacije.

V pravilniku ni opredeljeno kateri dokumenti morajo nastati znotraj vsakega projekta, je pa iz pravilnika mogoče razbrati, da mora dokumentacija projekta vsebovati vsaj interni vzpostavitveni dokument projekta in še nekaj dokumentov, ki so vezani na pogodbe in odnose z naročnikom (npr. pogodba, zapisniki sestankov, prevzemni zapisnik in podobno).

Informacijski sistem za podporo projektnemu vodenju v podjetju je v pravilniku opredeljen kot skupina aplikacij, ki so vzpostavljene v podjetju in jih je potrebno oz. se jih lahko uporablja pri upravljanju s projekti.

Izmed vseh naštetih aplikacij bi izpostavil le sistem »FogBugz«, ki služi za operativno spremljanje opravljenih ur na projektih in omogoča tudi definiranje posameznih nalog in vnos ocen, ter omogoča spremljanje porabe ur, virov in beleženje zahtev, napak in drugih parametrov projekta, ki nastajajo predvsem v fazi izvajanja projekta. Poleg aplikacije »FogBugz« pravilnik opredeljuje še nekatere druge interne aplikacije za npr. beleženje pogodb in referenc, podpori uporabnikom in podobno.

4 OPAZOVANA RAZVOJNA EKIPA

Razvojno ekipo, ki jo obravnavam v tej nalogi, sestavlja sedem članov. Od teh sedmih članov lahko pet oseb opredelimo kot razvijalce in dve osebi kot vodji projektov. Potrebno je upoštevati, da te osnovne vloge niso eksplicitno določene, saj lahko kateri od razvijalcev opravlja tudi vlogo vodje projekta na kakšnem primernem projektu. Prav tako niso vse naštete osebe tudi vezane samo na to ekipo, ampak lahko sodelujejo tudi na projektih drugih ekip, če imajo primerna znanja in če razmere v podjetju to zahtevajo. Osebi, ki sem ju opredelil kot vodji projektov, se od ostalih ločita predvsem po tem, da ne opravljata razvojnih del (programiranja), ampak predvsem vodita projekte in opravljata tudi kakšne druge naloge, ki ne spadajo med razvojne funkcije (npr. načrtovanje, zajemanje zahtev, predlaganje rešitev, pogajanja z naročniki in podobno). Prav tako ti dve osebi opravljata naloge vodij projektov tudi za druge ekipe v podjetju ali pa se ukvarjata še s katero drugo funkcijo v organizaciji (strateško, upravljalno, HRM, …).

Opazovana razvojna ekipa v podjetju pokriva predvsem znanja za razvoj programskih rešitev v Microsoft razvojnih okoljih. Pod razvojno kolje se v tem primeru štejejo tehnologije informacijske infrastrukture (aplikativni in podatkovni strežniki, operacijski sistemi), programsko razvojna okolja (razvojni okvirji (angl.: *Framework*)), razvojna orodja in programski jeziki ter sistemi za upravljanje podatkovnih zbirk.

Razvijamo programske rešitve, ki tečejo na aplikativnih strežnikih »Microsoft Internet Information Server« v okolju Microsoft Windows (operacijski sistem), kadar gre za večuporabniške strežniške rešitve. Kot razvojni okvir uporabljamo Microsoftov Framework .NET v navezi s programskima jezikoma C# in Visual Basic. Pod omenjeni razvojni okvir spada še množica drugih tehnologij, ki se jih uporablja. ASP.NET in ASPx za internetne strani in aplikacije ter MVC, LINQ, WinForms in še druge tehnologije razvojnega okolja .NET. Osrednje orodje za razvoj je Microsoft Visual Studio 2012 (in tudi starejše različice) v navezi s Team Foundation Server platformo za skupinski razvoj in verzioniranje izvorne kode.

Za podatkovni del naših programskih rešitev uporabljamo Microsoftovo rešitev za sistem podatkovnih zbirk »Microsoft SQL Server« in pripadajoče orodje »Microsoft Management Studio« ter poizvedbeni jezik T-SQL.

Poleg razvoja informacijskih rešitev na Microsoft platformi se ekipa v zadnjem času ukvarja tudi z razvojem aplikacij za mobilne naprave. Pri tem za enkrat pokrivamo tehnologije za razvoj aplikacij za Google Android platformo in platformo iOS podjetja Apple. Pri razvoju za

operacijski sistem Android se uporablja programski jezik Java v Android razvojnem okolju z orodjem Eclipse. Pri razvoju za operacijski sistem iOS pa smo uporabili orodje »MonoTouch«, ki omogoča razvoj iOS aplikacij z uporabo .NET programskih pristopov (programski jezik C#, LINQ, …).

Poleg naštetih orodij ekipa na projektih uporablja tudi druga orodja, ki so uvedena v podjetju. Med drugim tudi že omenjeni sistem za beleženje ur in planiranje nalog »FogBugz«, ter orodje »Balsamiq Mockups« za skiciranje in načrtovanje uporabniških vmesnikov.

V opazovani ekipi prevladujejo projekti izdelave programskih rešitev na zahtevo za sicer različne stranke, vendar te najpogosteje izhajajo iz slovenskega javnega sektorja oz. javne uprave. Pri teh projektih gre za izdelavo informacijskih rešitev za specifične potrebe posamezne stranke. Največkrat so namenjeni informacijski podpori strankinih organizacijskih funkcij ali procesov ter vodenju dokumentacije ali postopkov.

Projekti izdelave informacijskih rešitev na zahtevo so tudi na splošno najpogostejši v podjetju in jih zato izvajamo po že precej ustaljenem procesu. Že v fazi pridobivanja projekta je navadno potrebno ugotoviti osnovni nabor zahtev oz. funkcionalnosti, ki jih stranka zahteva, in na podlagi tega izdelati okvirno oceno potrebnega napora (ur dela). Ko je projekt pridobljen, se navadno izdela načrt končne rešitve, ki zajema opis funkcionalnosti, skice uporabniškega vmesnika in osnovni podatkovni model. Ob tem potekajo tudi usklajevanja in dogovarjanja s stranko zato, da se v načrtu zajamejo vse funkcionalnosti, ki jih stranka potrebuje na način, ki ji najbolj ustreza.

Izdelava načrta oz. specifikacije končne rešitve se včasih izvede tudi kot poseben projekt. Ta faza je uspešna zaključena takrat, ko imamo izdelan dovolj podroben načrt končne rešitve, ki ga stranka razume in se z njim tudi strinja. Šele ko je dosežen ta pogoj, se lahko prične faza implementacije oz. izdelave programske rešitve. V tej fazi se tudi določi natančnejše terminske okvirje za nadaljnje faze.

V fazi implementacije se sprogramirajo vse v načrtu specificirane funkcionalnosti, izvaja se testiranje in izdela se načrt za namestitev programske opreme v produkcijsko okolje. V tej fazi kakršnokoli spreminjanje načrtovanih funkcionalnosti predstavlja problem, zato se takšne spremembe obravnava po načelih upravljanja sprememb v metodologiji projektnega vodenja. Sprememba se primerno ovrednoti, ter se oceni, kakšen zamik v načrtovanih rokih predstavlja, ter kolikšen je potreben dodaten nabor dela. Nato se oceni, če je spremembo smotrno izvesti pod ugotovljenimi pogoji.

Vpogled naročnika v nastajajočo programsko rešitev v fazi implementacije se omogoči, če je bilo tako dogovorjeno v fazi načrtovanja, drugače pa se ob koncu implementacije izdelana programska oprema namesti v naročnikovo testno okolje in se mu ponudi čas, da programsko rešitev preizkusi in ugotovi morebitna odstopanja od specifikacij. V primeru pripomb naročnika se programsko rešitev ustrezno popravi in dopolni, dokler obe strani ne dosežeta dogovora, da je rešitev skladna s specifikacijo iz faze načrtovanja in tako ustrezna za namestitev v produkcijsko okolje naročnika.

Ekipa naročniku po namestitvi nove programske opreme zagotavlja še nekaj mesecev garancijske podpore za odpravo morebitnih napak. Navadno pa se sklene dogovor oz. pogodba za vzdrževanje, ki za dogovorjeno obdobje naročniku zagotavlja odpravo vseh napak, ki bi se pojavile v obsegu dogovorjenih mesečnih ur dela. Možen je tudi dogovor, da se naročniku nudi določeno število ur, ki se lahko porabijo za razvoj nadgradenj programske opreme.

Drug tip projektov, ki jih izvajamo v opazovani ekipi, so projekti razvoja produktov. Gre za manjši delež projektov, saj imamo trenutno v skupini razvit zgolj en večji produkt v obliki aplikacije za upravljanje z dokumenti. Programska rešitev je bila razvita že pred leti, sedaj pa se na tem projektu izvajajo predvsem odprave napak, nameščanja pri strankah, uporabniška podpora in pa občasne nadgradnje z novimi funkcionalnostmi. Med projekte razvoja produktov lahko štejem še razvoj aplikacij za mobilne naprave. Trenutno imamo razvito eno takšno aplikacijo za platformo Apple. Gre za projekt manjšega obsega.

Projekti razvoja produktov se od projektov izdelave rešitev po naročilu razlikujejo predvsem po tem, da naročnik projekta prihaja iz istega podjetja. Izoblikovanje zahtev za nove funkcionalnosti se torej izvaja znotraj podjetja in vključuje tudi člane opazovane ekipe. Vsako nadgradnjo se, podobno kot pri projektih po naročilu, najprej oceni, specificira v obliki načrta, implementira, preizkusi in nato namešča pri končnih uporabnikih. Faze se ustrezno poenostavijo, če gre za manjše spremembe ali dograditve.

Tretji tip projektov so projekti razvoja informacijskih rešitev za lastne potrebe. Gre za projekte razvoja programske opreme, ki jo podjetje uporablja za podporo lastnih funkcij oz. procesov poslovanja in vodenja projektov. Informacijska infrastruktura podjetja je sicer sestavljena iz množice aplikacij na različnih tehnologijah. Tako je le manjši del teh aplikacij razvit v opazovani ekipi. Ti projekti so navadno označeni z nižjimi prioritetami in se izvajajo takrat, ko se v ekipi pojavijo prosti viri. Funkcionalne zahteve za te programske rešitve definira vodstvo podjetja v sodelovanju z najpogostejšimi uporabniki posameznih aplikacij ali z lastniki procesa na katerega se nanaša aplikacija. Ker informacijsko okolje podjetja sestavljajo tudi kupljeni programski produkti, se izvajajo tudi dograditve teh produktov za lastne potrebe, če je to mogoče.

5 OCENA PRIMERNOSTI METODOLOGIJE SCRUM

Da bi lahko ugotovil, ali je uvedba metode Scrum v neko okolje smotrna in izvedljiva, moram izvesti sistematično ocenjevanje primernosti te metode. Predpostavim lahko, da je uvedba neke metode v ekipo smotrna v primeru, da je ocena prednosti, ki bi jih s tem ekipa pridobila, višja od ocene tveganja uvedbe in slabosti uvedbe te nove metode. Najprej bom poskusil našteti prednosti metode in jih v obliki kriterijev ovrednotiti tako, da bom lahko podal oceno koristnosti za opazovano ekipo. Naštel bom tudi pogoje, ki jih uvedba metode Scrum zahteva in jih prav tako oblikoval v kriterije, s katerimi bom lahko ovrednotil primernost opazovane ekipo za metodo Scrum. Na koncu bom s primerjanjem teh dveh ocen lahko določil, ali je uvedba metode Scrum v ekipo smotrna in izvedljiva.

5.1 Prednosti uporabe metode Scrum

V tem poglavju bom skušal našteti ugotovljene prednosti, ki jih prinaša metoda Scrum, in jih oblikovati v kriterije, s katerimi bom lahko nato izvedel ocenjevanje primernosti. Vsak kriterij bo obsegal točke od 0 do 5 in uteži od 0 do 100%. Višja ocena pri kriteriju prednosti pomeni, da ta prednost metode Scrum ocenjevani ekipi predstavlja višjo vrednost in odpravlja več problemov, s katerimi se srečuje ekipa sedaj. Pri vsakem kriteriju bom opisal, kaj pomeni najvišja in kaj najnižja ocena.

5.1.1 Učinkovitejše obvladovanje dokumentacije na projektu

Tradicionalne metode zahtevajo precej ukvarjanja z obsežnim dokumentiranjem razvojnih postopkov v okviru projektov izdelave informacijskih rešitev. Pogosto se dokumentacija uporablja kot sredstvo za komuniciranje in kot orodje za ugotavljanje pravilnosti izdelanih programskih rešitev. Ker zaporedno organiziran proces (npr. pri slapovni metodi) zahteva v prvih fazah izdelavo celotnega plana končne rešitve, mora biti ta zelo dobro dokumentiran, da si lahko skozi dokument načrta naročnik predstavlja, kaj bo v končni rešitvi dobil. Usklajevanje z naročnikovimi željami poteka navadno skozi dokument načrta nastajajoče programske opreme, ker pomeni, da je potrebno takšen dokument stalno popravljati in prilagajati naročnikovim željam. Specifikacija programske opreme mora biti dovolj podrobna, da se lahko z njeno pomočjo ob koncu projekta tudi oceni, če izdelana programska rešitev dejansko pokriva vse dogovorjene funkcionalnosti. Sestavljanje tako podrobne dokumentacije je zahtevno opravilo, ki porabi precej časa na projektu. Prav tako je ob vsaki spremembi, ki se pojavi v zahtevah v času planiranja ali celo kasneje, potrebno popravljati in voditi tudi vse dokumente, ki opisujejo in opredeljujejo končno rešitev.

Metoda Scrum sledi manifestu agilnega razvoja, ki med drugim postavlja delujočo programsko opremo pred vseobsežno dokumentacijo. Scrum to uresničuje z uvedbo precej točno opredeljene oblike artefaktov oz. dokumentacije, ki služijo spremljanju in dokumentiranju zahtev in funkcionalnosti nastajajoče programske opreme. Scrum pozna dva tipa dokumentacije načrta. To sta seznam zahtev produkta in seznam zahtev sprinta. Proces vzdrževanja teh dveh »dokumentov« pa je vpet v sam proces razvoja programske rešitve po Scrum metodologiji, ki točno definira, kdaj in kako se lahko vnašajo spremembe v načrt, ter katera vloga je za to odgovorna. Scrum na primer ne predvideva tako natančnega načrta, ki bi vseboval tudi skice celotnega uporabniškega vmesnika. Nasprotno. Za opis zahtev so dovolj uporabniške zgodbe, ki predstavljajo opis posamezne funkcionalnosti na kratek način skozi oči uporabnika. Veliko manj napora je potrebnega za dodajanje nove uporabniške zgode ali navesti spremembo obstoječe uporabniške zgodbe v seznamu zahtev izdelka, kot pa risanje novih ali popravljanje obstoječih skic uporabniškega vmesnika v natančnem dokumentu specifikacije. Prav tako ni dokument specifikacije tisti, ki služi predstavitvi končne programske rešitve naročniku. To bolje opravljajo izgotovljene izdaje programske opreme ob koncu vsakega sprinta.

Kriterij prednosti učinkovitejšega obvladovanja dokumentacije razvoja se z višjo oceno oceni pri ekipah, ki v obstoječem procesu porabijo za vodenje dokumentacije večji obseg celotnega časa trajanja projekta. Torej bodo ekipe, ki trenutno za vodenje dokumentacije porabijo najvišji delež celotnega napora, ta kriterij ocenile z več točkami.

5.1.2 Planiranje napora

Vsak projekt informacijske tehnologije zahteva tudi izvajanje planiranja potrebnega dela, časa, ljudi in drugih virov, ki so potrebni za doseganje cilja. Skozi planiranje se ugotavlja najprej smotrnost projekta, v nadaljevanju pa tudi potrebo po virih in določanje rokov posameznih faz, mejnikov in končanja celotnega projekta. Nenatančno planiranje prinaša težave, ki se kažejo v pogosti preobremenjenosti virov, hitenju na račun kvalitete, prekoračenju dogovorjenih rokov in v slabem strateškem usmerjanju projekta, ki ga izvajajo vodje projektov.

Tradicionalne metode, ki ne poznajo agilnih in iterativnih pristopov, zahtevajo dobre ocene potrebnega napora že v zgodnjih fazah projekta, saj se načrt programske opreme v celoti izdela na začetku. Takrat se določijo tudi roki za končno dobavo programske opreme, kar zahteva izdelavo ocene potrebnega napora v enem celovitem kosu za celoten projekt. To je naporna naloga in pogosto se dogaja, da so tako pridobljene ocene netočne (pogosto preveč optimistične). Osnova za ocenjevanje je specifikacija celotne programske opreme, ki na začetku še ni popolnoma usklajena z naročnikom, v samo ocenjevanje pa je vključen manjši nabor oseb. Dogaja se, da se pri takšnem ocenjevanju izpuščajo naloge (sploh manjše malenkosti), ki se razkrijejo šele tekom projekta. V skupno oceno napora pa je potrebno vključiti tudi izvajanje obrobnih aktivnosti in dogodkov na projektu, ki pa pri tradicionalnih metodah niso vnaprej definirane v samem procesu (sestanki, analiza novih zahtev, spremembe, pogajanja in podobno). Zato jih je tudi težje predvideti in oceniti.

Metodologija Scrum s svojim iterativnim procesom omogoča razvoj, ki je porazdeljen na manjše iteracije oz sprint-e, kar pomeni, da se tudi planiranje in ocenjevanje napora razbije na manjše koščke. Planiranje se tako izvaja ob začetku vsakega sprinta, pri tem pa se ocenjuje napor potreben za izvedbo posameznih uporabniških zgodb s seznama zahtev produkta, ki bodo uresničene v naslednjem sprintu. Takšno razbijanje večjega problema (ocena napora za celoten, še ne točno opredeljen, projekt) na manjše kose (uporabniške zgodbe enega sprinta) prinaša boljšo natančnost ocen in bolj predvidljiv napredek projekta.

Scrum metodologija sicer ne opredeljuje nobeno specifično tehniko ocenjevanja napora za posamezne predmete seznama zahtev izdelka, se pa pogosto, kot priporočena metoda ocenjevanja omenja tehnika poker planiranja. Poker planiranja predvideva vključitev večjega obsega ljudi (razvijalcev) v proces ocenjevanja in se zato v ocenah odražajo mnenja celotne ekipe. Tehnika omogoča, da razvijalci slišijo tudi mnenja in obrazložite drugih ocenjevalcev kar zmanjšuje možnosti, da se kakšna malenkost oz. skrite informacije pozabi upoštevati pri oddaji posamezne ocene in se tako poveča točnost ocen. Skupinsko odločanje o ocenah in povprečenje ocen, ki ju uvaja poker planiranja, prav tako pripomoreta k večji točnosti ocen. Z uvedbo uporabe tehnike poker planiranja pa naloga ocenjevanja za razvijalce postane manj

obremenjujoča. Ocenjevanje postane bolj zabavno in odgovornost za ocene se razporedi na celotno skupino razvijalcev.

Ker Scrum metoda za večino aktivnosti, ki niso del samega razvoja funkcionalnosti, (načrtovanje, planiranje, ocenjevanje, sestanki, …) opredeljuje posebne dogodke, ki so tudi postavljeni v točne časovne okvire, je planiranje in ocenjevanje potrebnega časa za te aktivnosti povsem nepotrebno, saj so definirane in časovno opredeljene že v samem Scrum procesu (npr. 4 urni okvir za izbiro predmetov s seznama zahtev izdelka v prvem delu sestanka za načrtovanje sprinta). Scrum proces zato že sam dovolj natančno napove, kolikšen čas in napor bo potreben za vse dodatne aktivnosti v sklopu posameznega sprinta in z vsakim dodatnim sprintom je čas za te aktivnosti že vnaprej znan.

Kriterij prednosti lažjega in natančnejšega planiranja se z višjo oceno oceni pri ekipah, ki se pogosteje srečujejo s težavami, ki jih prinašajo nenatančne ocene predvidenega napora, slabo ocenjujejo in planirajo svoje projekte in ne uporabljajo nobene posebne tehnike za pridobivanje ocen potrebnega napora.

5.1.3 Učinkovitost sestankov

Pri izdelavi specifikacij nastajajoče programske opreme in drugih aktivnostih, ki služijo oblikovanju končne rešitve in zbiranju uporabniških zahtev, je bistveno učinkovito sodelovanje in usklajevanje z naročnikom in njegovimi željami ter potrebami. Sodelovanje in usklajevanje z naročnikom, poleg komuniciranja po telefonu, elektronski pošti in skozi dokumentacijo, poteka tudi v obliki sestankov, ki se pogosto skličejo z namenom predstavitve načrtovanih rešitev in razjasnitve naročnikovih zahtev. Tradicionalni pristopi ne opredeljujejo števila takšnih srečanj, saj se ti navadno skličejo po potrebi. Prav tako ni vnaprej definirano, kdo vse se naj posameznega sestanka udeleži in kakšna je posameznikova vloga. Težave, ki pri tem nastajajo, se odražajo v predolgem izvajanju sestankov in v manjših rezultatih sestankov. Na sestankih se zbere neprimerno število udeležencev, ki nato v sklopu sestanka razpravljajo o težavah, ki ne spadajo v sklop projekta. Udeleženci, ki niso direktno zavezani k projektu, si vzamejo prevelik del sestanka in ga preusmerjajo na zadeve, ki niso bistvene za projekt. Pogosto se preveč detajlno razpravlja o stvareh, ki segajo predaleč v prihodnost, namesto da se razjasnjuje stvari, ki so trenutno »vroče« na projektu.

Scrum s svojimi dogodki in pravili zelo dobro in jasno opredeljuje, kaj se naj dogaja v sklopu posameznega sestanka, kdo se ga lahko udeleži in kakšna je njegova vloga, ter kdaj se naj sestanek zgodi in kako dolgo naj traja. Opredeljuje štiri dogodke, ki se izvajajo v obliki sestankov (sestanek za načrtovanje sprinta, dnevni Scrum, revizija sprinta in retrospektiva sprinta). Za vsak dogodek definira časovni okvir, s čimer je zagotovljeno, da se sestanki ne izvajajo predolgo. V Scrum pravilih je jasno opredeljeno o čem se na posameznem sestanku (ali delu sestanka) razpravlja in pa kdo oz. katere vloge se ga naj udeležijo. Z ločenjem ljudi med krovni vlogi »piščancev« in »prašičev« Scrum metoda omejuje možnost, da bi ljudje z manjšo zavezanostjo k projektu vodili sestanek, ali preveč posegali v njegov potek. Posamezni sestanki so v Scrum metodi tudi dobro vsebinsko opredeljeni, kar onemogoča, da

bi se na sestanku izgubljal čas z razpravljanjem o stvareh, ki niso vsebinsko povezane z namenom posameznega sestanka.

Kriterij učinkovitosti sestankov se z višjo oceno oceni pri ekipah, ki na svojih projektih neučinkovito obvladujejo sestanke, porabijo za njih preveč časa in imajo težave s pravilnim fokusiranjem vsebin sestankov. Ekipe z višjo oceno pri tem kriteriju bi z uporabo Scrum dogodkov pridobile največ koristi pri obvladovanju sestankov in srečanj na svojih projektih.

5.1.4 Pretok znanja

Znanje razvojne ekipe se nahaja v posameznikih, ki ekipo sestavljajo. Vsak posameznik je strokovnjak na enem ali več področjih in hkrati laik na drugem področju. Ker je za dosego ciljev projektov informacijske tehnologije potrebno znanje več področij, je za uspeh ključno, da ekipa poseduje vsa potrebna znanja, in da to znanje učinkovito prehaja med člani. Reševanje tehnološkega problema z vnovičnim iskanjem znanja, ki ga je nekdo v ekipi že osvojil, predstavlja neučinkovitost, saj se določeno delo opravlja dvakrat. Takšne situacije se ne dogajajo v ekipah, kjer je izmenjava znanja visoka in so člani ekipe med sabo seznanjeni o sposobnostih in izkušnjah drugih članov. Pogoj za tekoč pretok znanja v ekipi je učinkovitost in količina komuniciranja, saj se znanje med člani pretaka s pomočjo komunikacije. Dobra komunikacija tudi prispeva k seznanjenosti ekipe o izkušnjah in sposobnostih posameznih članov tako, da je vsakomur omogočeno, da ob potrebi hitreje poišče znanje, ki se v ekipi že nahaja.

Scrum spodbuja komunikacijo med člani samo-organizirane razvojne ekipe. Višji nivo komunikacije zagotavlja z rednimi vsakodnevnimi sestanki (dnevni Scrum sestanek) in s pristopom »face-to-face« komuniciranja. Dnevni Scrum dogodek, na katerem mora vsak član razvojne ekipe prispevati k skupni komunikaciji, je zelo dobra spodbuda za vse člane ekipe, da se jasno opredelijo o svojem delu na projektu in o problemih. Prav tako pa zagotavljanje samo-organiziranosti ekipe zahteva, da člani ekipe tesno sodelujejo, učinkovito komunicirajo in se seznanjajo o sposobnostih v ekipah tako, da si znajo sami najučinkoviteje razporediti delo glede na posameznikove sposobnosti in znanja.

Višja ocena za kriterij pretoka znanja se dodeli ekipam, ki se srečujejo s težavami, ki jih prinaša premajhna pretočnost znanja (npr. se pogosto raziskujejo tehnološke rešitve, ki so že bile osvojene ali pa člani ekipe težje poiščejo nekoga, ki poseduje neko specifično znanje v ekipi). Ekipe, ki so pri tem kriteriju višje ocenjene, bi z uvedbo metode Scrum več prispevale k boljši komunikaciji v ekipi.

5.1.5 Hitrost in učinkovitost razvoja

Iterativni pristop metode Scrum omogoča razdelitev celotnega projekta na manjše kose oz. iteracije (sprint). V času enega sprinta je ekipa tako osredotočena na realizacijo manjšega obsega uporabniških zgodb, kar prinaša boljšo obvladljivost in večjo osredotočenost ekipe. V času sprinta se vedno razvijajo tiste zahteve, ki naročniku trenutno predstavljajo najvišjo vrednost. Naročnik dobi ob koncu sprinta novo izgotovljeno različico programske opreme, ki

je jo je že mogoče vpeljati v uporabo. Razvoj v času enega sprinta poteka nemoteno, tudi če zahteve za nadaljnje iteracije še niso popolnoma definirane in stabilne. Samo-organizirana ekipa si naloge znotraj sprinta razporeja sama in pri tem upošteva izkušnje, znanja in motivacijo posameznih članov ekipe. Tako razporedi naloge bolj učinkovito. Ekipa je v času izvajanja sprinta stabilna, saj vanjo ni posegov, se ji ne dodaja ali odvzema članov in se ji ne diktira novih nalog ali načina izpolnjevanja izbranih uporabniških zgodb sprinta. Vsi ti našteti prijemi metode Scrum omogočajo hitrejši razvoj funkcionalnosti in oblikujejo okolje, ki je za delo ekipe najučinkovitejše.

Kriterij hitrosti in učinkovitosti razvoja se ocenjuje tako, da se ekipam, ki imajo težave z učinkovitim organiziranjem razvoja dodeli višja ocena. Večjo korist metode Scrum bodo torej imele ekipe, ki redno ne dosegajo dogovorjenih rokov, se dostikrat poslužujejo nadurnega dela razvijalcev in prerazporejanja virov in imajo na splošno težave z zagotavljanjem pogojev za učinkovito delo razvojne ekipe.

5.1.6 Primernost nove programske opreme za zahteve naročnika

Scrum metoda je, kot vse podobne agilne metode, naravnana predvsem k dobavi programske opreme, ki je kar najbolje prilagojena prav za naročnikove specifične zahteve in naročniku predstavlja najvišjo vrednost. Ker so spremembe del kompleksnega okolja v katerem nastajajo programske rešitve, je obvladovanje sprememb vpeto v sam proces razvoja po Scrum metodi. Pristop s krajšimi iteracijami naročnikom omogoča zgodnejše vpoglede v dejanske programske rešitve in tako zgodnejše proženje sprememb ter prilagajanje funkcionalnosti k dejanskim potrebam naročnika.

Pravilno nastavljanje prioritet predmetom s seznama zahtev izdelka prinaša to prednost, da so funkcionalnosti, ki za naročnika predstavljajo najmanjšo vrednost, postavljene vedno na konec seznama, s tem pa dosežemo, da so v tekočem razvoju samo tiste stvari, ki so ob začetku sprinta bile označene z višjimi prioritetami. To pomeni, da se prioritetno razvija res samo tisto, kar naročnik najbolj potrebuje. Tako se tudi zmanjša razvoj funkcionalnosti, ki so manj pomembne ali celo nepotrebne. Naročnik ima ob koncu vsakega sprinta tudi možnost ustaviti nadaljnji razvoj, če ugotovi, da do sedaj razvita programska oprema že pokriva vse njegove potrebe. Prav tako pa ima možnost ob koncu sprinta spremeniti zahteve, jih drugače oblikovati in tako usmerjati razvoj v smer, ki mu prinaša najvišjo vrednost.

Takšen proces razvoja zagotavlja, da je končna izdelana programska oprema primernejša za naročnika, bolje pokriva njegove zahteve in mu prinaša višjo vrednost.

Kriterij primernosti programske opreme za naročnika se oceni tako, da se več točk dodeli tistim ekipam, ki sedaj ne uspevajo s svojim razvojem najbolje zadovoljiti vse naročnikove zahteve in programsko rešitev bolje prilagoditi njegovim specifičnim potrebam. Naročniki ne zaznavajo visoke vrednosti v dobljeni programski opremi in po prejemu pogosteje takoj zahtevajo izvedbe nadgradenj, sprememb in drugih prilagajanj dobavljenih informacijskih rešitev.

5.1.7 Nastavljanje uteži kriterijem prednosti

Vsem zbranim kriterijem, ki opisujejo prednosti metode Scrum, bom nastavil še uteži. Z utežmi želim kriterije ovrednotiti tako, da bo pri končni oceni kriterij, ki je pomembnejši oz. je prednost, ki jo opisuje, več vredna, tudi bolj vplival na končno oceno. Torej se kriteriju, ki opisuje prednost Scruma z večjo vrednostjo, dodeli višja utež. Uteži se nastavlja v odstotkih tako, da je seštevek vseh uteži vseh kriterijev enak 100 %. Vsak kriterij nosi tako le delež celotnega nabora uteži.

Pri nastavljanju uteži sem si pomagal s člankom avtorjev Urevc in Mahnič »Ocena prednosti metode Scrum in njenih tipičnih praks« (Urevc & Mahnič, 2012, str. 189), kjer sta avtorja uspela z anketiranjem uporabnikov metode Scrum izmeriti, kako uporabniki cenijo posamezno prednost Scruma oz. katere prednosti se v praksi dejansko najbolje izkažejo.

Ugotovila sta, da uporabniki metode Scrum vse njene prednosti ocenjujejo kot koristne. Še najbolj cenjeni pa sta boljša obvladljivost projekta zaradi krajših iteracij in izboljšana komunikacija med člani razvojne skupine (Urevc & Mahnič, 2012, str. 189). Upoštevajoč te ugotovitve bom višje uteži zato nastavil pri kriterijema »Hitrost in učinkovitost razvoja« ter »Pretok znanja«.

Tabela 2: Tabela uteži za kriterije prednosti

KRITERIJ	RAZLAGA	UTEŽ(%)
Učinkovitejše obvladovanje dokumentacije na projektu	Prve tri kriterije ocenjujem kot enakovredne in jim nastavim enake uteži glede na preostanek od najbolj izpostavljenih zadnjih treh kriterijev.	11,0
Planiranje napora		11,0
Učinkovitost sestankov		11,0
Pretok znanja	Pretok znanja, kot posledica izboljšane komunikacije, je ena izmed prednosti, ki so jo uporabniki metode Scrum tudi bolj izpostavili (Urevc & Mahnič, 2012, str. 189).	22,0
Hitrost in učinkovitost razvoja	Boljša obvladljivost zaradi iterativnega pristopa omogoča višjo učinkovitost razvoja. To je tudi pri uporabnikih najbolj cenjena prednost metode Scrum (Urevc & Mahnič, 2012, str. 189).	25,0
Primernost nove programske opreme za zahteve naročnika	Vrednost programske opreme za naročnika je ena od bistvenih prednosti, ki jih želimo doseči z uporabo agilnih metodologij, in je opisana že v samem manifestu agilnega razvoja (»**Delujoča programska oprema** pred vseobsežno doku..«). Utež se zato nastavi nekoliko višje.	20,0

5.2 Kriteriji primernosti razvojne ekipe za uvedbo metode Scrum

V naslednjih točkah bom poskusil našteti pogoje oz. ovire, ki se lahko pojavijo pri uvajanju metode Scrum v neko razvojno ekipo in pri izvajanju projektov po Scrum metodi.

Iskanje takšnih pogojev v obstoječi literaturi je težavno, saj se večina literature ukvarja s faktorji uspeha na projektih, ki se že upravljajo s katero od agilnih metodologij. Torej ti kriteriji uspeha pridejo v poštev v organizaciji, ki že deluje agilno. Chow in Cao (2007, str. 963) tako na primer naštejeta kar 36 kriterijev uspeha agilnih projektov, ki jih razporedita v štiri kategorije: organizacija, ljudje, procesi, tehnologija in projekti.

O ovirah pri sprejemanju metode Scrum sta se razpisala tudi Leffingwell in Smits (2005, str. 15). Pri tem ugotavljata, da je vse pogoje in ovire za takšen prehod nemogoče napovedati vnaprej. Šele ko se Scrum metoda dejansko začne uporabljati v ekipi, se bodo prave ovire in težave prikazale. Vseeno sta uspela napovedati nekaj splošnih ovir, na katere je potrebno biti pripravljen, preden se začne metoda Scrum vpeljevati v razvojno ekipo. Večina navedenih problemov se zopet nanaša na primere, ko se Scrum praksa že izvaja. Nekaj pa jih lahko uporabimo za oceno primernosti opazovane ekipe za metodo Scrum.

Z preučevanjem še drugih virov in literature sem uspel izbrati nekaj ključnih faktorjev, ki vplivajo na uspeh vpeljave agilne metode, in jih lahko uporabim za podajo ocene primernosti metode Scrum za opazovano razvojno ekipo.

Vse pogoje bom, enako kot v prejšnjem poglavju, oblikoval v kriterije, ki bodo omogočili izvedbo ocenjevanja. Vsak kriterij vsebuje nabor ocen od 0 do 5 in uteži od 0 do 100%. Pri kriterijih primernosti višja ocena predstavlja višjo pripravljenost ekipe na posamezen kriterij. Torej višjo oceno kot ekipa prejme, lažje bo uvedla in uporabljala metodo Scrum pri svojem delu.

5.2.1 Izkušenost razvijalcev

Razvijalci, ki imajo več izkušenj s področja razvoja in tehnologije (npr. obvladajo večje število programskih jezikov, poznajo več procesov razvoja in se že dlje časa ukvarjajo s tem področjem), so bolj sposobni razumeti in obvladovati znanja o agilnih metodologijah in lažje prepoznajo prednosti agilnega razvoja. Izkušnje tudi pripomorejo k lažji uporabi agilnih metod (Chan & Thong, 2008, str. 808).

Ekipa se za kriterij izkušenosti razvijalcev oceni tako, da se večje število točk dodeli ekipam, ki jih sestavljajo izkušenejši razvijalci. Šteje se število poznanih programskih jezikov, obseg poznanih procesov razvoja, leta izkušenj in drugi kriteriji, ki nakazujejo posameznikovo izkušenost na področju razvoja informacijskih rešitev.

5.2.2 Urjenje

Urjenje pripomore k povečanju zmožnosti učenja agilnih pristopov in pri prenosu

teoretičnega znanja v prakso. Uvedba tehnoloških inovacij vedno prinaša tudi določeno mero negotovosti, ki jo je z urjenjem mogoče zmanjšati. Urjenje tako izboljšuje zmožnost sprejemanja agilnih metodologij (Chan & Thong, 2008, str. 808).

Višje število točk za ta kriterij se dodeli ekipam, ki so že izvajale kakršnokoli obliko urjenja uporabe ali uvedbe metode Scrum ali vsaj katerega od njenih pristopov. Prav tako se več točk dodeli ekipam, ki imajo vsaj možnost takšno urjenje izvesti v prihodnosti.

5.2.3 Zunanja pomoč

Pri uvajanju nove agilne metode je priporočljivo poiskati zunanjo pomoč v obliki eksperta, saj navadno v ciljni organizaciji ni mogoče najti dovolj izkušeno osebo za učinkovito implementacijo nove metode v ekipo. Zunanji svetovalec zna v podjetje prinesti svež pogled na razvoj in poenostaviti prehod na agilno metodo (Chan & Thong, 2008, str. 809).

Če je zunanja pomoč dosegljiva in jo je mogoče izrabiti v polni meri, se ekipi za ta kriterij dodeli višje število točk.

5.2.4 Podpora vodstva

Organizacije, v katerih je vodilni menedžment bolj naklonjen k posodabljanju tehnološke infrastrukture, so tudi bolje pripravljene za inovacije. Podpora vodstva je pomemben faktor pri implementaciji agilnih metodologij v okolje. Takšna podpora predstavlja dobro spodbudo in motivacijo za uporabo agilnih metod in uveljavitvi agilnih pristopov (Chan & Thong, 2008, str. 810). Poleg tega pa popolna podpora vodstva lahko predstavlja razvijalcem dejstvo, da je uporaba agilne metode Scrum in njenih pristopov tudi edina pravilna pot pri razvoju in pogoj, da je projekt lahko označen kot uspešen s strani vodilnega menedžmenta.

Ekipam, katerim vodstvo nudi višjo podporo za uvedbo nove metode, se za omenjeni kriterij dodeli višje število točk. Prav tako šteje tudi angažiranost vodstva v promoviranje uporabe agilnih pristopov metode Scrum in druge smernice zaposlenim, ki lahko prihajajo s strani vodilnega menedžmenta.

5.2.5 Organizacijska kultura

Organizacijska kultura je pomemben faktor, ki lahko vpliva na sprejemljivost razvojne metode. Pomanjkanje kulture znanja predstavlja glavno oviro za uspešno upravljanje z znanjem v programskem inženirstvu. Kot kulturo znanja razumemo okolje, ki ponuja priložnosti za ustvarjanje znanja, ter spodbuja učenje in deljenje znanja znotraj organizacije. V takšnih okoljih oz. kulturi, kjer se poudarja pomembnost širjenja znanja in spodbuja upravljanje znanja, je sprejemljivost agilnih metod višja (Chan & Thong, 2008, str. 810).

Kriterij organizacijske kulture se oceni tako, da se višje število točk dodeli v primeru, da je v organizacijskem okolju opazovane razvojne ekipe zaznati boljše pogoje za izmenjavo in pretok znanja, ter se širjenju, pridobivanju in vzdrževanju znanja namenja več pozornosti. K

boljši oceni pripomorejo organizirani notranji tečaji in predstavitve, udeleževanje zunanjih izobraževanj in tehnoloških konferenc in urejeni odnosi v organizaciji, ki pripomorejo k večji izmenjavi izkušenj med sodelavci.

5.2.6 Sposobnost dela v timu

Pripravljenosti posameznika, da deluje oz. dela kot član ene ali več ekip, pravimo sposobnost dela v timu. Delo v timu povečuje spodobnost sprejetja novih tehnologij. Timsko delo in tesno sodelovanje je tudi zahteva v vseh agilnih metodah za doseganje višje učinkovitosti. Timsko delo zmanjšuje psihološke in fizične razdalje med ljudmi in ustvarja pogoje za spodbujanje učenja in prehajanje znanja med zaposlenimi. Dobro timsko delo povečuje zmožnost posameznikov, da sprejmejo principe agilnih metod (Chan & Thong, 2008, str. 810).

Kriterij sposobnosti dela v timu se bolje oceni pri ekipah, ki že delujejo v utečenih timih in vsebujejo posameznike, ki se v timu znajdejo in znajo dobro funkcionirati znotraj ekip. Štejejo komunikativnost in pripadnost ekipi, ter zmožnost sprejemanja in izražanja idej.

5.2.7 Komuniciranje

Podobno kot sposobnost dela v timu je tudi komuniciranje tisto, ki pripomore boljšemu pretoku znanja med razvijalci in omogoča boljšo uporabo agilnih metod. Predvsem pogosta in natančna govorna komunikacija med člani ekipe pripomore dosegati boljšo učinkovitost (Chan & Thong, 2008, str. 810).

Kriterij komuniciranja se z več točkami oceni pri ekipah, pri katerih je zaznati višji nivo komunikacije. Razvijalci med seboj pogosteje izmenjavajo mnenja in dajejo nasvete, ter si izmenjujejo izkušnje. Še bolje je, če je kakšen vidik komuniciranja tudi bolj formalno urejen (npr. z vnaprej določenimi sestanki ali srečanji namenjenimi izmenjavi izkušenj in idej).

5.2.8 Razmerje z naročnikom

Dobro razmerje oz. razumevanje med razvijalci in naročnikom olajša pretok interpretacij in znanja o zahtevah. Ker je sodelovanje naročnika del Scrum procesa razvoja, je vzdrževanje dobrega odnosa z naročnikom pomemben faktor za uspešno vpeljavo agilne metode (Chan & Thong, 2008, str. 811).

Boljša ocena se za kriterij razmerij z naročniki dodeli ekipam, ki s svojimi naročniki že sedaj zelo dobro sodelujejo, komunicirajo in nimajo težav pri pridobivanju potrebnih informacij s strani naročnikov.

5.2.9 Zaznana uporabnost metode

Ta kriterij se nanaša na samo metodo Scrum. Ugotovljeno je, da je ob večji zaznani uporabnosti metode, večja tudi njena sprejemljivost. To pomeni, da bodo posamezni člani

ekipe lažje sprejeli delo po novi metodi, če bodo zaznali njeno dejansko uporabnost in njen doprinos k učinkovitejšem delu (Chan & Thong, 2008, str. 811).

Ekipi, kateri člani višje vrednotijo uporabnost metode Scrum, se za ta kriterij dodeli tudi višje število točk. Če se večina članov ekipa strinja, da je uporabnost Scrum metode visoka, bo ta kriterij ocenjen z višjo oceno.

5.2.10 Zaznana preprostost uporabe metode

S tem kriterijem označimo stopnjo, po kateri posameznik meni, da uporaba neke rešitve/metode ne bo zahtevala večjih umskih naporov. Ker prehod iz tradicionalnih metod razvoja na agilne lahko predstavlja tudi večje spremembe v procesih, lahko zato preprostost dojemanja agilnih principov izbrane metode pomeni tudi lažje sprejemanje takšne metode s strani članov ekipe (Chan & Thong, 2008, str. 811).

Za kriterij se torej dodeli višje število točk ekipam, kateri člani nimajo težav z razumevanjem pristopov in principov metode Scrum in so tudi mnenja, da so zmožni sprejeti, razumeti in delovati po njenih usmeritvah.

5.2.11 Zaznana kompatibilnost metode

Faktor označuje, koliko je nova metoda konsistentna (podobna) z obstoječo metodo, procesom ali organiziranostjo razvoja. Večja zaznana kompatibilnost pomeni, da bi pri uvajanju nove agilne metode bilo potrebno uveljaviti manj sprememb v proces razvoja. Razvijalci bodo veliko lažje sprejeli metodologijo, ki je bolj kompatibilna z obstoječim stanjem (Chan & Thong, 2008, str. 811).

Za kriterij kompatibilnosti metode se z več točkami oceni ekipe, ki že sedaj uporabljajo več agilnih pristopov pri svojem delovanju in je njihova obstoječa metoda tako bližja metodi Scrum. Več kot je že uveljavljenih agilnih pristopov, lažji bo prehod na metodo Scrum in več točk se ekipi lahko dodeli za ta kriterij.

5.2.12 Prihod članov ekipe prepozno na dnevni Scrum sestanek

Dnevni Scrum sestanek je del Scrum procesa. Pravila Scrum-a so zelo jasna. Dnevni Scrum se mora zgoditi vsak delovni dan ob isti uri in na istem kraju. Udeležijo naj se ga vsi člani razvojne ekipe. Priporočljivo je, da je to prva stvar v delovnem dnevu ekipe. Zamujanje na ta dogodek torej ni primerno, povzroča težave v procesu, ter omejuje komunikacijo in pretok znanja.

Določeno ekipo lahko po tem kriteriju ocenimo le s predvidevanjem, ali bi opazovana ekipa lahko redno in dovolj učinkovito izvajala dnevne Scrum sestanke. Višja ocena se dodeli ekipi, ki ima za izvajanje teh srečanj omogočene tudi vse pogoje.

5.2.13 Prekinjanje dela posameznih članov z namenom, da se jih napoti na izvajanje drugih nalog izven tekočega sprinta

Med izvajanjem posameznega sprinta je kakršnokoli zunanje poseganje v razvojno ekipo nezaželeno. Ekipa mora skozi sprint namreč doseči cilje sprinta, ki so bili zastavljeni na sestanku za načrtovanje sprinta. Seznam zahtev sprinta se planira tako, da ga ekipa lahko v celoti izvede v času enega sprinta, pri tem je upoštevana hitrost razvojne ekipe in njen ritem. Če med izvajanjem sprinta razvijalce prekinjamo ali jih razvrščamo na druge naloge, lahko to povzroči, da se poruši ritem, ki ga želi ekipa vzdrževati skozi celoten sprint, cilj sprinta pa lahko postane nedosegljiv. Poleg tega Scrum metoda predvideva samo-organiziranost razvojne ekipe, in so tudi zato zunanji posegi vanjo nepriporočljivi.

Kriterij prekinjanja dela se ocenjuje glede na sposobnost ekipe in njenega okolja, da razvijalcem zagotavlja nemoteno delo v okviru izvajanj sprint-ov. Če se v obstoječi praksi izvajanja projektov pogosto dogajajo posegi v delo razvijalcev in njihovo preusmerjanje na druge naloge, se takšni ekipi dodeli manj točk za ta kriterij.

5.2.14 V ekipi obstajajo viri za vse zahtevane funkcionalnosti projekta (načrtovanje, oblikovanje, programiranje, testiranje, …)

Scrum razvojne ekipe so sestavljene tako, da njihovi člani pokrivajo vsa znanja in spretnosti, ki jih zahteva posamezen projekt. To predvsem vključuje znanja o načrtovanju in oblikovanju rešitev, znanja za implementacijo rešitev (programiranje, podatkovne baze, …) in spretnosti za testiranje programske opreme. Pričakuje se, da ekipa v svoji sestavi vsebuje vsa potrebna znanja in spretnosti za izvedbo projektov in ni veliko potreb po zunanji pomoči in drugih ukrepih, ki bi ovirali vzdrževanja utečenega ritma dela ekipe in njene samo-organiziranosti.

S tem kriterijem se ocenjuje celovitost opazovane ekipe in njeno pokritost z znanji. Ekipe, ki za svoje projekte že posedujejo vsa potrebna znanja in ne iščejo veliko zunanje pomoči, se ocenijo z višjim številom točk.

5.2.15 Problem projektov s fiksno ceno in fiksnimi datumi

Scrum metodologija je zelo neprimerna za projekte, na katerih so obsegi stroškov in časovni obsegi določeni vnaprej in fiksni (vpisani v pogodbo). Pichler (2010, str. 78) priporoča, da se takšnih projektov izogibamo. Schwaber (2004, str. 134) prav tako ugotavlja, da Scrum metoda ni primerna za takšne projekte. Metodo Scrum označi kot »umetnost mogočega« in ne kot »daj mi tisto, kar sem plačal in to takrat, ko si rekel, da bo dostavljeno«. Za Scrum so torej veliko primernejše odprte pogodbe, kjer se dopuščajo spremembe, ki se med razvojem programske opreme zagotovo pojavljajo. Le tako je mogoče uveljaviti vse prednosti metode Scrum in zagotoviti pravo vrednost programske opreme za naročnika ob pravem času.

Kriterij torej ocenjuje primernost pogodb oz. primernost odnosov z naročniki opazovane ekipe za metodologijo Scrum. Ekipe, ki imajo s svojimi naročniki dovolj dobre odnose, da

lahko izvajajo projekte, ki niso že takoj na začetku omejeni s ceno in roki, bodo lažje uveljavile Scrum pristop. Takšne ekipe se zato pri tem kriteriju tudi bolje ocenijo.

5.2.16 Nastavljanje uteži kriterijem primernosti

Tudi kriterijem primernosti bom nastavil ustrezne uteži tako, da bodo kriteriji, ki predstavljajo večjo oviro pri uvajanju in uporabi metode Scrum, bolj vplivali ne oceno. Takšni kriteriji bodo imeli zato višjo utež.

Tabela 3: Tabela uteži za kriterije primernosti

KRITERIJ	RAZLAGA	UTEŽ(%)
Izkušenost razvijalcev	Kriterij ima manjši vpliv na primernost ekipe.	4,5
Urjenje	Kriterij ima manjši vpliv na primernost ekipe.	4,5
Zunanja pomoč	Kriterij ima manjši vpliv na primernost ekipe.	4,5
Podpora vodstva	Podpora vodstva precej vpliva na uspeh vpeljave in uporabe novih metod, zato temu kriteriju dodelimo večjo utež.	12,5
Organizacijska kultura	Kriterij ima srednji vpliv na primernost ekipe.	6,0
Sposobnost dela v timu	Kriterij ima srednji vpliv na primernost ekipe.	6,0
Komuniciranje	Kriterij ima srednji vpliv na primernost ekipe.	6,0
Razmerje z naročnikom	Kriterij ima srednji vpliv na primernost ekipe.	6,0
Zaznana uporabnost metode	Kriterij ima manjši vpliv na primernost ekipe.	4,5
Zaznana preprostost uporabe metode	Kriterij ima manjši vpliv na primernost ekipe.	4,5
Zaznana kompatibilnost metode	Kriterij ima srednji vpliv na primernost ekipe.	6,0
Prihod članov ekipe prepozno na dnevni Scrum sestanek	Kriterij ima srednji vpliv na primernost ekipe.	6,0
Prekinjanje dela posameznih članov z namenom, da se jih napoti na izvajanje drugih nalog izven tekočega sprinta	Kriterij ima večji vpliv na primernost ekipe.	8,0
V ekipi obstajajo viri za vse zahtevane funkcionalnosti projekta	Kriterij ima srednji vpliv na primernost ekipe.	6,0
Problem projektov s fiksno ceno in fiksnimi datumi	Različni avtorji enako opozarjajo na neprimernost pogodb s fiksnimi cenami in datumi za Scrum metodo, zato temu kriteriju določim najvišjo utež.	15,0
Prihod članov ekipe prepozno na dnevni Scrum sestanek	Kriterij ima srednji vpliv na primernost ekipe.	6,0

Uteži mi bodo tudi omogočile, da bom lažje primerjal točke prednosti in točke primernosti pri podajanju končne ocene. Uteži vseh kriterijev skupaj namreč znašajo 100 %. To je enako pri obeh skupinah kriterijev, kar mi omogoča, da lahko enakovredno primerjam zbrane točke obeh skupin kriterijev. Pri obeh skupinah kriterijev je, z upoštevanjem uteži, možno nabrati največ 5 točk skupaj iz vseh kriterijev. Pri končni oceni se namreč točke posameznega kriterija pomnožijo z vrednostjo uteži.

6 OCENA PRIMERNOSTI METODE SCRUM ZA OPAZOVANO EKIPO

V tem poglavju bom opazovano ekipo ocenil po vseh zbranih kriterijih. Na podlagi izbranih točk bom podal oceno primernosti in smiselnosti uvedbe nove metodologije Scrum v opazovano ekipo.

6.1 Ocena opazovane ekipe po kriterijih prednosti uporabe metode Scrum

Najprej ocenim opazovano ekipo po kriterijih prednosti metode Scrum. Pri vsakem kriteriju bom podal tudi razlago, ki opisuje odločitev za izbrano število točk.

- **KRITERIJ: Učinkovitejše obvladovanje dokumentacije na projektu:**

  V opazovani razvojni ekipi se projekti razvoja izvajajo po precej zaporedni metodi, kar pomeni, da se v prvih fazah planiranja izdela natančen načrt oz. specifikacija končne rešitve. Ker spremembe v kasnejših fazah niso zaželene, mora ta specifikacija zelo dobro odražati točno kaj in kako bo razvito v končni rešitvi, da bodo strankine zahteve in pričakovanja kar najbolje zadovoljene. Specifikacija obsega opise vseh funkcionalnosti, načrt osnovnega podatkovnega modela, diagram primerov uporabe (angl.: *Use Case Diagram*) in skice celotnega uporabniškega vmesnika. Ker se v fazi načrtovanja izvaja tudi usklajevanje z naročnikom, je dokument načrta tisti, ki naročniku skuša razjasniti, kako bo na koncu izgledala in delovala končna programska rešitev. Dokument je tudi osnova za ocenjevanje popolnosti izdelane programske opreme ob prevzemu. Ob prevzemu nove programske opreme mora ta vsebovati vse, kar je v dokumentu načrta navedeno in nobena dodatna funkcionalnost se ne more zahtevati, če ni navedena v načrtu ali pa bila vanj dodana naknadno ob sprejeti spremembi. Ker se dokument uporablja tudi kot orodje za predstavitev načrtovane programske opreme naročniku v fazi planiranja in usklajevanja z naročnikom, je v tej fazi podvržen tudi spreminjanju, dopolnjevanju in preoblikovanju, kar pa seveda vzame precej časa in napora, saj je včasih potrebno popraviti skoraj večino skic uporabniškega vmesnika, ko naročnik izrazi kakšno novo željo ali spremembo. Te nove želje ali spremembe pa se navadno porajajo, ko naročnik vidi dovolj podrobno skico uporabniškega vmesnika planirane programske rešitve.

  Ocenim lahko, da opazovana ekipa za obvladovanje osnovne dokumentacije specifikacije projekta sedaj porabi precej več napora, kot pa bi ga z uporabo Scrum pristopov.

OCENA: 4

- **KRITERIJ: Planiranje napora:**

  V opazovani ekipi se ob fazi pridobivanja projekta in v fazi načrtovanja izvaja tudi ocenjevanje potrebnega napora in planiranje izvedbe razvoja nove programske rešitve. Ocene napora zahteva vodja projekta, saj jih potrebuje za izvedbo svojih aktivnosti v prvih fazah projekta. Samo ocenjevanje izvajajo razvijalci na projektu, pri tem pa se ne uporablja nobena specifično določena metoda ali pristop za podajo teh ocen. Razvijalcu je lahko v pomoč popis tehnik ocenjevanja, ki ga je izdelal in predstavil eden od sodelavcev v podjetju. Ocenjevanje je precej nezaželena aktivnost, predstavlja naporno nalogo, ki brez prave metode sloni dostikrat na golem ugibanju. Poleg tega pa prinaša določeno mero odgovornosti, saj se od razvijalca pričakuje, da bo za svojimi ocenami stal, ko se bo programska rešitev razvijala. Razvijalci se največkrat poslužujemo tehnike razgradnje tako, da se zahteve razbijejo na čim manjše koščke, ki jih je potem lažje oceniti. Ocenjujeta navadno eden ali dva razvijalca. Občasno pa se za mnenje o ocenah vpraša še druge razvijalce s podobnimi znanji. Ocene se vedno podajajo v obliki števila ur, ki so predvidene za razvoj posamezne funkcionalnosti. Kadar je ocene potrebno podati že v fazi pridobivanja projekta, je to še toliko težja naloga, saj še niso razjasnjene vse zahteve ali funkcionalnosti pričakovane programske rešitve, oceniti pa je potrebno tudi druge aktivnosti na projektu, npr. izdelava načrta programske rešitve.

  Tako pridobljene ocene napora so dostikrat netočne, navadno preveč optimistične in planiranje projekta zato sloni na napačnih predvidevanjih. V opazovani ekipi sicer zaradi tega naročniki navadno ne trpijo zamujanj, je pa nenatančno planiranje zlasti moteče za vodje projektov, ki zato težje razporejajo vire, planirajo nadaljnje projekte in aktivnosti ter vrednotijo finančni vidik projekta. Z drobljenjem projektov po iteracijah in vpeljavo boljše metode ocenjevanja napora bi ekipa pridobila na natančnejših ocenah napora in lažjim planiranjem dela na svojih projektih.

  OCENA: 3

- **KRITERIJ: Učinkovitost sestankov:**

  V opazovani ekipi se sestanki na projektih izvajajo različno učinkovito, kar je predvsem odvisno tudi od naročnika in njegove ekipe, ki je zadolžena za projekt. Pri naročnikih, s katerimi se sodeluje že daljši čas in na več projektih, se sestanki in dogovarjanja izvajajo že precej utečeno in brez pretirane porabe časa, medtem ko se pri novih naročnikih pogosto dogajajo težave, ki so opisane v tem kriteriju (neprimerno število udeležencev, razpravljanje izven teme sestankov, ...). Obvladovanje srečanj po Scrum principu in po vnaprej določenih dogodkih bi povečalo učinkovitost predvsem na projektih z novimi naročniki.

  OCENA: 2

- **KRITERIJ: Pretok znanja:**

  Izmenjava znanja v opazovani razvojni ekipi ni upravljana po kakšni metodi ali pristopu. Vso znanje ekipe se načeloma odraža v znanju posameznih članov, vendar se vsaka novo pridobljena veščina vedno ne prenaša tudi na ostale člane ekipe oz. se dogaja, da se člani ekipe med sabo ne seznanjajo dovolj dobro o novo pridobljenih znanjih. Člani ekipe sicer pogosto komuniciramo med sabo in se zanimamo za delo drugih članov ter se tako seznanjamo o njihovem znanju. Prav tako se ob soočanju s kakšnim tehnološkim problemom pogosto za mnenja sprašuje tudi druge sodelavce, vendar to ni pravilo niti ustaljena praksa. Takšno izmenjavanje znanja je odvisno od posameznikovih osebnostnih lastnosti (komunikativnost, smisel za timsko delo, …). Podjetje skuša spodbujati pretok znanja z različnimi pristopi. Omogoča ter spodbuja izvedbe predstavitev oz. notranjih izobraževanj, kjer imajo posamezniki priložnost, da vse zainteresirane seznanijo s tehnologijo ali drugim znanjem, ki so ga med delom na novo osvojili ali kako drugače obvladali. Vzpostavljena je tudi manjša aplikacija, ki služi kot zbirka znanja za pisno beleženje znanj in tehnik, vendar je slabo uporabljena.

  Opisano stanje sicer ni zelo problematično, saj je nekaj postopkov za izmenjavo znanja že vzpostavljenih. Vseeno se dogaja, kar vsekakor predstavlja problem in neučinkovitost, da se posamezni razvijalec ukvarja z reševanjem kakšnega tehnološkega problema, ki ga je v ekipi pred časom že uspešno razrešil drug član ekipe.
  OCENA: 2

- **KRITERIJ: Hitrost in učinkovitost razvoja:**

  Po metodah projektnega vodenja je vodja projekta tisti, ki koordinira delo na projektu. Takšen pristop predvideva, da vodja projekta določa prioritete nalog, njihov vrstni red in izbira ljudi, ki bodo posamezno nalogo realizirali. Znotraj opazovane razvojne ekipe sicer takšno strogo izvajanje praks projektnega vodenja ni uveljavljeno, a je vseeno vodja projekta tisti, ki ima vodilno vlogo pri koordiniranju dela na projektu. Ekipi razvijalcev na projektu se dopušča, da si delo sami organizirajo oz. so aktivno vključeni v organiziranost. Vseeno je poseg v delo ekipe pogost, predvsem takrat, ko se pojavijo potrebe po razvijalcih še na drugih projektih in se posameznike razporeja po projektih med samim izvajanjem nalog. Delo ekipe se pogosto tudi prekinja zaradi kakšnih nalog (predvsem vzdrževalnih) na drugih projektih ali pa s kakšnimi drugimi aktivnostmi (izobraževanja, sestanki, podajanje mnenj ipd.). Vse to vpliva na hitrost razvoja in onemogoča, da bi ekipa vzpostavila določen ritem dela, ki ga poznamo iz metode Scrum.

  Opazovana ekipa lahko pričakuje višjo učinkovitost razvoja z uporabo Scrum pristopov, ki predvsem zagotavlja boljše pogoje za delo razvojne ekipe in vzpostavlja okolje, ki je optimalno za njeno samo-organiziranost, motivacijo in stabilnost.
  OCENA: 5

- **KRITERIJ: Primernost nove programske opreme za zahteve naročnika:**

  V fazi načrtovanja se, po ustaljeni metodi dela v opazovani razvojni ekipi, izdela podroben načrt končne programske opreme. V načrtu se skuša karseda dobro opisati vse funkcionalnosti, ki bi najbolje zadovoljile zahteve naročnika. Osnova za izdelavo tega načrta so ugotovljene uporabniške zahteve, analiza obstoječih rešitev, analiza naročnikove organizacije ter procesov, usklajevanja z naročnikom, ter drugi dosegljivi vsebinski artefakti, ki pomagajo pri oblikovanju končne programske rešitve. Ključno pri oblikovanju načrta je naročnikovo sodelovanje in zahteve, ki nam jih v tem sodelovanju uspe ugotoviti. Ker se v tej fazi nobena funkcionalnost dejansko še ne razvije, ampak obstaja zgolj na papirju (v obliki opisov in skic uporabniškega vmesnika), si naročnik vedno ne zna dovolj dobro predstavljati kako točno bo stvar videti v končni obliki in če bo zadovoljila njegove zahteve in pričakovanja. Naročniki pogosto šele ob dejanskem preizkušanju programske opreme oz. ob njeni uporabi v realnem okolju sprevidijo, kako dejansko ta deluje in koliko rešitev pokriva njihove zahteve oz. jim zagotavlja tisto, kar so z razvojem dejansko želeli doseči. Direktno preizkušanje končane programske opreme generira tudi največ zahtev po spremembah s strani naročnika, saj mu dejanska uporaba aplikacij sproža nove ideje za njeno uporabo in oblikuje nove zahteve. Vsekakor lahko opazimo, da izgotovljena programska oprema v prvih verzijah na projektih opazovane ekipe vedno najbolje ne zadovoljuje potreb in pričakovanj naročnika. Zahteve po nadgradnjah takoj po prvi namestitvi so stalnica na vseh projektih. Pogosto se dogaja tudi, da se razvijejo funkcionalnosti, za katere se nato z uporabo izkaže, da jih naročnik niti ne potrebuje, ali pa vsaj ne v tako velikem obsegu, kot so implementirane v končni rešitvi.

  OCENA: 4

Opazovana razvojna ekipa je torej zbrala naslednje točke za kriterije prednosti uporabe Scrum metode:

Tabela 4: Izračun ocene prednosti uporabe metode Scrum

KRITERIJ	OCENA	UTEŽ (%)	KONČNA OCENA UPOŠTEVAJOČ UTEŽI
Učinkovitejše obvladovanje dokumentacije na projektu	4	11,0	0,44
Planiranje napora	3	11,0	0,33
Učinkovitost sestankov	2	11,0	0,22
Pretok znanja	2	22,0	0,44
Hitrost in učinkovitost razvoja	5	25,0	1,25
Primernost nove programske opreme za zahteve naročnika	4	20,0	0,80
		SKUPAJ:	**3,48**

Skupna ocena za kriterije prednosti je seštevek vseh že oteženih ocen. Ta za opazovano ekipo znaša **3,48 točk**. Poenostavljeno povedano bi torej opazovana ekipa z uvedbo metode Scrum pridobila za 3,48 točk izboljšav, ki jih prinaša takšna metoda. Če upoštevamo, da je po moji metodi ocenjevanja možno izbrati največ 5 točk za kriterije prednosti, lahko ocenim, da bi opazovana ekipa lahko precej napredovala in izboljšala svoj proces z uporabo nove agilne metode.

6.2 Ocena opazovane ekipe po kriterijih primernosti ekipe za uvedbo metode Scrum

V tem poglavju opazovano ekipo ocenim še po kriterijih primernosti za uvedbo metode Scrum. Pri vsakem kriteriju bom podal tudi razlago, ki opisuje odločitev za izbrano število točk.

- **KRITERIJ: Izkušenost razvijalcev:**

  Vsak razvijalec opazovane razvojne ekipe ima najmanj 4 leta izkušenj v razvoju programskih rešitev in vsi obvladamo vsaj tri različne programske jezike. Izkušnje z različnimi metodami in procesi razvoja so večinoma omejene zgolj na proces razvoja v podjetju (projektno vodenje – slapovni pristop) in na izkušnje z metodami, ki so jih nekateri uporabljali v drugih podjetjih, kjer so bili zaposleni prej. Z agilnimi pristopi smo vsi vsaj delno seznanjeni, saj so posamezni razvijalci o takšnem pristopu poslušali najmanj eno predstavitev na kateri od tehnoloških konferenc, ki se jih občasno udeležujemo. Ocenjujem lahko, da je izkušenost razvijalcev zadovoljiva, zato ne bi smela predstavljati težav pri prehodu na agilno metodo Scrum.
  OCENA: 4

- **KRITERIJ: Urjenje:**

  Opazovana razvojna ekipa do sedaj še ni imela priložnosti, da bi izvedla kakšen poskusni projekt po metodi Scrum, zato o izkušnjah z metodološkim urjenjem ne moremo govoriti. Vsekakor pa je to lahko eden od možnih korakov pri izvedbi uvajanja nove metode razvoja v ekipo.
  OCENA: 2

- **KRITERIJ: Zunanja pomoč**

  Zunanja pomoč za uvedbo metodologije Scrum v opazovani ekipi ni na voljo, vendar v podjetju že imamo pozitivne izkušnje z najemanjem zunanjih svetovalcev pri uvajanju novih procesov poslovanja in organizacije. Če bi bila podpora vodstva za uvedbo Scrum metodologije dovolj velika, bi lahko razmišljali tudi o takšnem najemu dodatne pomoči za uvajanje nove metode. V Sloveniji je na trgu tudi mogoče najti ustrezne podjetja, ki se ukvarjajo s svetovanjem o agilnem razvoju in metodi Scrum.
  OCENA: 2

- **KRITERIJ: Podpora vodstva:**

  V opazovanem podjetju vodstvo navadno podpira inovativne pristope za izboljšanje tako procesa razvoja kot samih programskih rešitev, vendar je ta podpora navadno premalo resno izražena, kar v praksi pomeni, da se mora ekipa sama motivirati v smeri uvedbe sprememb v proces razvoja. Zelo pomembna je pri tem tudi vloga projektnih vodij, ki bi s strani vodstva morali dobiti jasno izraženo zahtevo, da se začnejo dosledno uporabljati principi metode Scrum na posameznih projektih. Podporo vodstva lahko v tem trenutku ocenimo zgolj v obliki dopuščanja ekipi, da samoiniciativno spremeni svoj proces razvoja. Zgolj takšno odobravanje pa je seveda premalo, da bi lahko govorili o sto odstotni podpori vodstva, ki bi aktivno stalo za to idejo in jo promoviralo v vse nivoje organizacije (razvijalcev, projektnih vodij in vodilnega menedžmenta).
  OCENA: 3

- **KRITERIJ: Organizacijska kultura**

  V podjetju se v okviru upravljanja z znanjem izvajajo aktivnosti internih in zunanjih izobraževanj. Z udeležbo konferenc, predavanj in delavnic se v obliki zunanjih izobraževanj pridobivajo nova znanja. Nova znanja se ustvarjajo tudi predvsem z deli na projektih, kjer se razvijalci samoiniciativno učijo novih tehnologij, da lahko razvijajo primerne programske rešitve. Pridobljena znanja se deli med druge zaposlene formalno preko notranjih predstavitev in neformalno preko sodelovanja na projektih. Okolje tako, po mojih ocenah, ponuja dovolj organizacijske kulture za sprejemljivost agilne metode Scrum, vsekakor pa na tem področju še obstaja prostor za izboljšave. Nekaj izboljšav bi privedla že sama Scrum metoda s svojimi principi deljenja znanj in izkušenj znotraj razvojne ekipe.
  OCENA: 5

- **KRITERIJ: Sposobnost dela v timu:**

  V opazovani ekipi je timsko delo že zelo dobro uveljavljen način sodelovanja na projektih. Vsi projekti v podjetju se izvajajo znotraj ekip, ki jih sestavljata najmanj dva razvijalca in projektni vodja, tako da nobeno delo ni nikoli samostojno vezano zgolj na enega razvijalca. V opazovani ekipi smo tako že dobro vajeni medsebojno sodelovati na projektih, izmenjavati znanje in izkušnje ter na splošno delovati kot tim.
  OCENA: 5

- **KRITERIJ: Komuniciranje:**

  Člani opazovane ekipe med sabo dobro komunicirajo v smislu neformalne verbalne komunikacije med delom na projektih. Sprožilci komuniciranja so navadno tehnološki problemi ali vsebinska vprašanja, ki nastajajo med razvojem funkcionalnosti posamezne programske opreme. Bolj formalno urejeno komuniciranje, kot ga uvaja metoda Scrum (npr. dnevni Scrum sestanki), bi stanje še izboljšalo, saj bi bilo uvedenih več sprožilcev komuniciranja, ki bi prispevali k pogostejši komunikaciji ter pretoku znanja in izkušenj.

Podjetje opazovane razvojne ekipe dovoljuje tudi, da zaposleni svoje delo opravljajo od doma. Člani razvojne ekipe se takšnega načina dela tudi redno poslužujemo. Delo do doma je precej uveljavljena praksa, ki zaposlenim omogoča, da ne zapravljajo časa z vožnjo na sedež podjetja (sploh pri sodelavcih, ki so bolj oddaljeni), da si delovni čas lažje prilagodijo svojim navadam ter življenjskim potrebam in si omogočijo delo v okolju, ki jim bolj ugaja (npr. izven hrupnih pisarn, ki so za nekatere moteče). Uporabljajo se uveljavljene tehnične rešitve za takšno opravljanje dela (oddaljen dostop do namizja, navidezna zasebna omrežja, aplikacije za takojšnje sporočanje, …). Seveda je vsem v ekipi jasno, da je vestno, kvalitetno in pravočasno opravljanje nalog prioriteta in merilo uspešnosti, in da je takšno delo omogočeno zaradi nivoja zaupanja, ki je vzpostavljeno v podjetju in ki ga je potrebno tudi vzdrževati. Pri uveljavljeni metodi opravljanja projektov takšen način dela tudi ne moti procesa projektnega vodenja, saj so naloge navadno vnaprej dodeljene in jih lahko posamezni člani precej samostojno opravljajo tudi od doma. Vsak član ekipe, ki svoje delo opravlja tudi od doma, je vseeno vsaj enkrat tedensko prisoten tudi na sedežu podjetja. Po potrebi pa je te prisotnosti še več (dogovorjeni sestanki, narava dela in drugi primerni razlogi, ko razvijalec oceni, da je za rešitev naloge bolje, da je prisoten v podjetju), zato nivo komunikacije ni bistveno oviran zaradi takšne lokacijske porazdelitve razvijalcev.

OCENA: 3

- **KRITERIJ: Razmerje z naročnikom:**

Z našimi naročniki navadno vzdržujemo dobre odnose, ki ne povzročajo težav v komunikaciji. Boljši odnosi se vzpostavijo predvsem pri naročnikih, s katerimi sodelujemo že dlje časa in smo z njimi sodelovali že na več projektih. Redko pa je sodelovanje z naročnikom tako intenzivno, kot to predvideva metoda Scrum.

OCENA: 2

- **KRITERIJ: Zaznana uporabnost metode:**

Uporabnost metode Scrum za opazovano razvojno ekipo sem ocenil z oceno prednosti, ki je relativno visoka. Pri trenutnem kriteriju pa je važno tudi, da takšno visoko uporabnost metode Scrum razumejo vsi člani ekipe in ostali sodelujoči, ki bi bili vpeti v njeno uvedbo. Tu seveda prava zaznavnost uporabnosti metode Scrum še ni dosežena, saj metoda še ni bila dovolj dobro predstavljena v okolju, njene prednosti pa še niso dovolj dobro promovirane.

OCENA: 2

- **KRITERIJ: Zaznana preprostost uporabe metode:**

Metoda Scrum je s svojimi vlogami, procesi, praksami, artefakti in nekaj pravili lahko v osnovi precej enostavno razumljiva in preprosta (Schwaber, 2004, str. 10). Ocenim lahko, da v opazovani ekipi nebi našli člana, ki bi ne bil sposoben preprosto dojeti procesa, vlog in pravil Scrum-a. Zaželena bi seveda bila primerna predstavitev metode in konkretna vloga skrbnika metodologije pri prvih projektih.

OCENA: 4

- **KRITERIJ: Zaznana kompatibilnost metode:**

  Obstoječ proces vodenja projektov po principih projektnega vodenja je precej tradicionalna metoda, ki se tudi dosti razlikuje od metode Scrum, saj ne predvideva uporabo večine agilnih principov. Nekaj principov agilnosti vseeno lahko zasledimo (uporaba objektnih programskih jezikov, standardi kodiranja in avtomatizirani testi), vendar je to premalo, da bi lahko obstoječe stanje ocenil kot kompatibilno z metodo Scrum. Obseg sprememb, ki bi jih bilo potrebno uveljaviti v proces, je torej nekoliko večji.
  OCENA: 2

- **KRITERIJ: Prihod članov ekipe prepozno na dnevni Scrum sestanek:**

  V opazovani ekipi oz. v podjetju je uveljavljen prilagodljiv delovni čas, kar pomeni, da zaposleni prihajajo na delovno mesto ob nekoliko različnih urah. To bi lahko oviralo redno izvajanje dnevnega Scrum sestanka. Vseeno ocenjujem, da to nebi smela biti ovira, saj se kljub prilagodljivemu delavniku vsi člani razvojne ekipe na svojem delovnem mestu pojavijo najkasneje do devete ure, sprejet pa je tudi dogovor, da se za dogovorjena srečanja in sestanke prihaja točno. Ob skupno dogovorjeni in sprejeti uri za izvedbo dnevnih Scrum sestankov torej v opazovani ekipi nebi smelo prihajati do zamujanj.
  OCENA: 4

- **KRITERIJ: Prekinjanje dela posameznih članov z namenom, da se jih napoti na izvajanje drugih nalog izven tekočega sprinta:**

  Ta pogoj v trenutnih razmerah okolja opazovane ekipe ni izpolnjen. Pogosto se namreč dogaja, da se posameznega razvijalca prekinja pri njegovem delu zaradi nalog na drugih projektih. Tu gre večinoma za vzdrževalne posege na rešitvah, ki so že nameščene pri naročnikih, ali pa pomoč pri razvoju ali planiranju na drugih projektih v podjetju. Za uspešno vpeljavo metode Scrum bi bilo potrebno uvesti nekaj sprememb, ali pa pri planiranju posameznega sprinta upoštevati, da je razpoložljivost razvijalcev na projektih lahko manjša od pričakovane.
  OCENA: 2

- **KRITERIJ: V ekipi obstajajo viri za vse zahtevane funkcionalnosti projekta:**

  Opazovana ekipa že sedaj večinoma samostojno izvaja vse naloge, ki jih zahtevajo projekti, tako da lahko ocenim, da ekipa vsebuje vse potrebne vire za izvedbo funkcionalnosti, ki jih lahko pričakujemo na prihajajočih projektih.
  OCENA: 4

- **KRITERIJ: Problem projektov s fiksno ceno in fiksnimi datumi:**

  V opazovani ekipi je večina projektov ravno takšne narave, da so stroški projekta in datum dostave zahtevane programske opreme določeni vnaprej in skoraj vedno vpisani tudi v pogodbo z naročnikom. Torej lahko ocenim, da gre večinoma za projekte s fiksno ceno in fiksnimi datumi ter stroški. To je še posebej značilno za projekte, kjer je naročnik državna uprava ali druga z državno upravo povezana organizacija, kjer se naročanje izvaja po sistemu javnih naročil, in je vnaprej napovedan obseg stroškov ključen. Problem s fiksnimi pogodbami sicer ne bi mogel označiti kot problem opazovane ekipe, vendar je pravilen odnos in sodelovanje z naročnikom, ki v tem primeru pogojuje obliko sodelovanja, ključen pogoj za uspešno uporabo Scrum metodologije.
  OCENA: 2

Opazovana razvojna ekipa je torej dosegla naslednje točke za kriterije primernosti razvojne ekipe za uvedbo metode Scrum:

Tabela 5: Izračun ocene primernosti ekipe za vpeljavo metode Scrum

KRITERIJ	OCENA	UTEŽ (%)	KONČNA OCENA UPOŠTEVAJOČ UTEŽI
Izkušenost razvijalcev	4	4,5	0,180
Urjenje	2	4,5	0,090
Zunanja pomoč	2	4,5	0,090
Podpora vodstva	3	12,5	0,375
Organizacijska kultura	5	6,0	0,300
Sposobnost dela v timu	5	6,0	0,300
Komuniciranje	3	6,0	0,180
Razmerje z naročnikom	2	6,0	0,120
Zaznana uporabnost metode	2	4,5	0,090
Zaznana preprostost uporabe metode	4	4,5	0,180
Zaznana kompatibilnost metode	2	6,0	0,120
Prihod članov ekipe prepozno na dnevni Scrum sestanek	4	6,0	0,240
Prekinjanje dela posameznih članov z namenom, da se jih napoti na izvajanje drugih nalog izven tekočega sprinta	2	8,0	0,160
V ekipi obstajajo viri za vse zahtevane funkcionalnosti projekta	4	6,0	0,240
Problem projektov s fiksno ceno in fiksnimi datumi	2	15,0	0,300
		SKUPAJ:	**2,965**

Skupna ocena za kriterije primernosti je seštevek vseh že oteženih ocen. Ta za opazovano ekipo znaša **2,965 točke**. Poenostavljeno povedano bi torej za opazovano ekipo ocenili, da je

s 2,965 točkami pripravljena za uvedbo metode Scrum. Če upoštevam, da je po moji metodi ocenjevanja možno zbrati največ 5 točk za kriterije primernosti opazovane ekipe, lahko ocenim, da je ekipa sicer dobro pripravljena, a vsekakor ne najbolje. Do idealne pripravljenosti (5 točk) ekipi manjka še 2,035 točk.

6.3 Končna ocena

Po opravljenih obeh ocenjevanjih opazovane ekipe, je končno odločitev o primernosti možno podati precej enostavno. Če zgolj primerjam obe pridobljeni oceni, vidim, da je zbranih točk prednosti 3,48, kar je več kot pa je točk, ki ekipi manjkajo do idealne pripravljenosti za uvedbo metode Scrum (2,035). Točke, ki ekipi manjkajo do idealne primernosti, lahko razumemo kot točke ovir za uvedbo nove metode, in teh je manj, kot pa je točk izboljšav, ki bi jih ekipa pridobila s takšno uvedbo. Ekipa bi torej z uvedbo te metode pridobila več izboljšav, kot pa obstaja ovir za njeno uvedbo. S to ugotovitvijo lahko ocenim, da bi bila uvedba metode smotrna in smiselna.

Vseeno je potrebno razmisliti o možnih izboljšavah v obstoječem okolju in opazovani ekipi, ki bi takšno uvedbo nove metode naredile še lažje izvedljivo in smotrnejšo in bi oceno primernosti bolj približala idealni vrednosti pripravljenosti ekipe. Predvsem sta zaskrbljujoči oceni pri najbolj izpostavljenima kriterijema primernosti. To sta podpora vodstva in problem projektov s fiksno ceno in fiksnimi datumi. Kljub pozitivno izmerjeni končni oceni za uvedbo metode Scrum, bi nizka ocena teh dveh kriterijev povzročala precej problemov pri uvedbi nove metode.

Pred pričetkom aktivnosti za uvajanje Scrum principov v ekipo bi bilo potrebno o smotrnosti prepričati vodstvo in pridobiti njegovo popolno podporo ter zagotovilo, da bo Scrum metoda postala standard in pravilo za izvajanje projektov v ekipi ali celo v celotni organizaciji. Vodstvo bi na ta način lahko to odločitev bolje promoviralo v vse nivoje organizacije, opravljeno delo pa bi se začelo vrednotiti tudi glede na upoštevanje principov metode Scrum.

Glede problema s pogodbami s fiksno ceno in datumi se nekaj priporočil najde tudi v literaturi. Tako na primer Schwaber (2004, str. 135), kot goreč zagovornik in soavtor metode Scrum, predlaga, da se v takšnih situacijah deloma projekt vseeno lahko vodi po nekaterih principih metode Scrum. Predlaga predvsem pravilno uporabo seznama zahtev izdelka (predvsem pravilna razporeditev prioritet predmetov in njihovo izvajanje v takšnem vrstnem redu) in pa uporabo mesečnih iteracij, kjer se ob koncu vsakega sprinta naročniku prikaže izdelano različico programske opreme. Na tak način želi naročnikom prikazati prednosti iterativnega pristopa in jih prepričati v bolj odprte pogodbe in tesnejše odnose, ki bi pripeljali do možnosti za celovitejšo uporabo metode Scrum. Pogoj za to pa je seveda ustrezna pripravljenost naročnika za razumevanje različnih metod vodenja projektov.

Kriterij komuniciranja bi lahko bil, zaradi omenjene lokacijske porazdelitve razvijalcev v ekipi zaradi uveljavljenega dela od doma, ocenjen tudi nižje. Vendar ker je tudi v mnogih drugih podjetjih že uveljavljeno delo od doma, v večjih korporacijah pa se projekti izvajajo celo v ekipah, ki so popolnoma ločene in porazdeljene po celem svetu, se je tudi za takšne

pogoje našla rešitev in uskladitev z metodo Scrum. V literaturi tako lahko zasledimo pojem porazdeljenega Scrum-a (angl.: *Distributed Scrum*). Woodward, Surek in Genis (2010) v svoji knjigi (»A Practical Guide to Distributed Scrum«) opišejo priporočila, kako izvajati projekte po Scrum metodi v primerih, ko je ekipa lokacijsko porazdeljena. Priporočila se nanašajo na večje projekte, kjer so člani locirani po vseh delih sveta (tudi po različnih časovnih pasovih, kar predstavlja še dodaten problem). Za opazovano ekipo sem lahko zato ocenil, da občasna porazdeljenost njenih članov ne bi smela predstavljati problema. Sprejet bi moral biti dogovor, da se ob večjih Scrum dogodkih (retrospektiva sprinta, sestanek za planiranje sprinta, …) vsi člani ekipe zberejo na sedežu podjetja. Pereč problem z izvedbo dnevnih Scrum sestankov pa je rešljiv z uporabo telefonskih konferenc, video konferenc ali z uporabo aplikacij za takojšnje sporočanje znotraj skupin (angl.: *Group Instant Messaging*) (Woodward, Surdek & Ganis, 2010, str. 103).

Nekoliko problematično je tudi prekinjanje dela posameznih članov z namenom, da se jih napoti na izvajanje drugih nalog izven tekočega sprinta (13. kriterij). Tu je mogoče predlagati kakšno rešitev v smislu oblikovanja ekip, ki nebi vsebovale ljudi, ki opravljajo tudi na primer naloge vzdrževanja. Za te naloge bi bil začasno zadolžen lahko samo en član ekipe, drugi pa bi bili polno zaposleni na sprint-ih. Mogoče je tudi posamezne člane samo delno planirati za vsak sprint posebej. Tako se ob pričetku vsakega sprinta oceni, koliko točk uporabniških zgodb bo posamezen član dejansko na razpolago (zaradi drugih nalog izven projekta). Nato pa se mu na podlagi tega primerno dodeli manjši nabor manj kompleksnejših nalog tekočega sprinta.

SKLEP

Scrum metodo razvoja informacijskih rešitev uvrščamo med agilne metodologije razvoja. Agilne metodologije so se pojavile zaradi zahteve po razvoju, ki bi bil bolj prilagojen na spreminjajoča in kompleksna okolja, v katerih danes nastajajo informacijske rešitve. Kompleksnost okolij prihaja iz tehnologij in orodij, razpoložljivosti strokovnjakov, metod, znanj, zahtevanih funkcionalnosti, konkurence, časovnih in finančnih omejitev ter ostalih faktorjev, ki vplivajo na razvoj posamezne programske opreme.

Agilne metodologije se kompleksnosti razvoja lotevajo z iterativnim in evolucijskim pristopom. Lastnosti agilnih metod se najbolje predstavi z manifestom agilnega razvoja: (Beck et al., 2001):

»Odkrivamo boljše načine razvoja programske opreme, tako, da jo razvijamo, in pri tem pomagamo tudi drugim. Naše vrednote so ob tem postale:

Posamezniki in interakcije pred procesi in orodji.
Delujoča programska oprema pred vseobsežno dokumentacijo.
Sodelovanje s stranko pred pogodbenimi pogajanji.
Odziv na spremembe pred togim sledenjem načrtom.

Z drugimi besedami, četudi cenimo dejavnike na desni, vseeno bolj cenimo tiste na levi.«

Scrum agilna metodologija predstavlja procesni okvir znotraj katerega je mogoče uporabiti različne procese ali tehnike. Scrum okvir sestavljajo vloge, dogodki, artefakti in pravila. Pravila Scruma povezujejo dogodke, vloge in artefakte, ter urejajo odnose in razmerja med njimi.
Scrum vpeljuje tri vloge: skrbnik metodologije, predstavnik naročnika in razvojna ekipa. Vsaka vloga ima svoj prispevek k projektu in svoj nabor zadolžitev oz. vpliva v potek projekta. Skrbnik metodologije skrbi, da delo in procesi sledijo Scrum principom. Predstavnik naročnika skrbi za nabor zahtev in želja naročnika in njihovo predstavitev razvojni ekipi, ta pa je zadolžena, da v samo-organizaciji izdela zahtevano programsko opremo na najučinkovitejši način in tudi tako, da bo končni produkt predstavljal čim višjo vrednost za naročnika.

Metodologija Scrum svoj proces obravnava skozi Scrum dogodke. Ti dogodki so: sestanek za načrtovanje sprinta, dnevni Scrum sestanek, sprint, revizija sprinta in retrospektiva sprinta. Kot produkt aktivnosti na Scrum dogodkih nastajajo Scrum artefakti. Scrum vpeljuje naslednje artefakte procesa: seznam zahtev izdelka, seznam zahtev sprinta, graf preostalega časa in definicijo "dokončanega".

Opazovana razvojna ekipa trenutno izvaja svoje projekte po metodi, ki še najbolj spominja na slapovni model, kjer si faze projekta sledijo ena za drugo. Podjetje, v katerem deluje opazovana razvojna ekipa, uporablja za vodenje projektov principe projektnega vodenja, ki so opredeljeni v poslovniku poslovanja.

Opazovana ekipa svoje projekte izvaja dokaj uspešno, vendar je v procesu razvoja možno zaznati tudi probleme, ki bi jih z uvedbo bolj agilnega pristopa lahko odpravili ali vsaj omilili. Dobro vpeljana metoda Scrum bi predvsem izboljšala organiziranost same ekipe in pohitrila dobavo programske opreme, ki bi bila bolje prilagojena naročnikovim potrebam. Zmanjšan pa bi bil tudi obseg dela na funkcionalnostih, ki se na koncu izkažejo kot nepotrebne. Scrum artefakti (npr. seznam zahtev izdelka) in principi upravljanja z njimi omogočajo, da je dokumentacija specifikacije nastajajoče programske opreme bolje vodena in bolj prilagojena stalnim spremembam v zahtevah, ki se pojavljajo v vseh fazah projekta, kar opazovani ekipi zdaj še predstavlja problem, saj se preveč časa namenja vzdrževanju specifikacij. Pri teh prednostih, ki jih prinaša metoda Scrum, je bila opazovana ekipa tudi najbolje ocenjena, kar pomeni, da bi te prednosti prinesle ekipi največ vrednosti ob uvedbi metode Scrum.

Izboljšave, ki bi jih opazovana ekipa še lahko pridobila z uvedbo nove metode, so:

- spodbuda za kakovostnejšo komunikacijo, ki bi dvignila nivo pretoka znanja v ekipi;
- bolje definirane vloge oseb in vsebine posameznih sestankov, s čimer ti postanejo bolj učinkoviti in časovno omejeni;
- uporaba boljših pristopov k planiranju in ocenjevanju posameznih nalog (npr. uporaba pokra planiranja). To bi v opazovani ekipi izboljšalo točnost ocen napora za naloge in proces ocenjevanja naredila manj obremenjujoč za razvijalce.

Pri ocenjevanju primernosti metode Scrum za opazovano ekipo in pri ocenjevanju pripravljenosti ekipe in njenega okolja na prehod na novo metodo sem ugotovil, da ekipa ocenjeno zelo dobro zadovoljuje pogoje izkušenosti razvijalcev, njihove sposobnosti dela v timu, organizacijske kulture, zaznane preprostosti uporabe metode in da v ekipi najdemo vse veščine za izvedbo celotnih projektov. Opazovana ekipo lahko dobro ocenim pri pogojih, ki se dotikajo obsega in kakovosti komunikacije ter podpore vodstva za uvedbo nove metode. Slabše pa se ekipa izkaže pri kriterijih urjenja in zunanje pomoči, kriteriju razmerij z naročniki (bolje sicer pri že obstoječih naročnikih), kriteriju seznanjenosti z uporabnostjo metode in pri kriteriju kompatibilnosti metode z obstoječim stanjem. Kot pereč problem se kažeta tudi praksa prerazporejanja članov med projekti in oblika pogodb z naročniki, ki so večinoma definirane s fiksnimi cenami in fiksnimi datumi.

Ugotovljena ocena, ki sem jo pridobil s primerjanjem ocene prednosti metode za opazovano ekipo in ocene primernosti ekipa za uvedbo metode Scrum, sicer pokaže, da bi bila uvedba metode Scrum v opazovano ekipo smiselna. Prednosti, ki bi jih s tem ekipa pridobila, je namreč več, kot pa je ovir za njeno uvedbo. Vendar je pred odločitvijo za novo metodo vseeno priporočljivo izvesti nekaj izboljšav v pogojih, da bi bila uvedba lažje izvedljiva in učinkovitejša.

Primerna bi bila večja podpora in angažiranost vodstva. Primerno pa bi bilo tudi doseči boljše odnose z naročniki, ki bi morali najprej pristati na bolj odprte pogodbe, kjer se cena in časovni obseg projekta ne omejita že na začetku, nato pa biti pripravljeni tudi na bolj intenzivno obliko sodelovanja na projektih, kot to veleva agilni princip Scrum metodologije.

Scrum metodologija torej za določene ekipe zagotovo prinaša večjo učinkovitost v proces razvoja programske opreme in informacijskih rešitev, kar lahko ocenimo tudi za primer opazovane ekipe. Ta bi Scrum metodo lahko vpeljala v svoje okolje, vendar bi zato bilo priporočljivo pridobiti večjo podporo vodstva in tudi pridobiti stranke, ki bi bile bolj pripravljene na projektih sodelovati po agilnem iterativnem principu Scrum metodologije.

LITERATURA IN VIRI

1. Ambler, S. (2008, februar). Agile Adoption Rate Survey Results. Najdeno 15. marec 2013 na spletnem naslovu http://www.ambysoft.com/surveys/agileFebruary2008.html.
2. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., C. Martin, R., Mellor, M., Schwaber, K., Sutherland, J., & Thomas, D. (2001). Manifest agilnega razvoja programske opreme. Najdeno 15. Marec 2013 na spletnem naslovu http://agilemanifesto.org/iso/sl/
3. Blankenship, J, Bussa, M., & Millett, S. (2011). *Pro Agile .NET Development with SCRUM* (1th ed.). New York: Apress
4. Blom, M. (2010). Is Scrum and XP siutable for CSE Development?. *Procedia Computer Science, 1*(1), 1511-1517.
5. Cervone, F. (2011). Understanding agile project management methods using Scrum. *OCLC Systems & Services, 27*(1), 18-22.
6. Chan, F., & Thong, J. (2008). Acceptance of agile methodologies: A critical review and conceptual framework. *Decision Support Systems. 4*(46), 803–814.
7. Chandra, Misra, S., Kumar, V., & Kumar, U. (2009). Identifying some important success factors in adopting agile software development practices. *Journal of Systems and Software, 11*(82), 1869-1890.
8. Chow, T., & Cao, D. (2007). A survey study of critical success factors in agile software projects. *The Journal of System and Software, 6*(81), 961–971.
9. Comland d.o.o. (2012). *Pravilnik o vodenju projektov podjetja Comland*. Ljubljana: Comland d.o.o.
10. Dingsoyra, T., Nerur, S., Balijepally, V., Brede, & Moe, N. (2012). A decade of agilemethodologies: Towards explaining agile software development. *The Journal of Systems and Software 6*(85), 1213-122.
11. Eduardo, M., & Bourque, P. (2010). Agile monitoring using the line of balance. *The Journal of Systems and Software, 7*(83), 1205–1215.
12. James, M. (2010) Scrum Reference Card. Najdeno 25. april na spletnem naslovu http://ScrumReferenceCard.com.
13. Laanti, M., Salo, O., & Abrahamsson. P. (2010). Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation. *Information and Software Technology, 3*(53), 276–290.
14. Leffingwell, D., & Smits., H. (2005). A CIOís Playbook for Adopting the Scrum Method of Achieving Software Agility. *Rally Software Development Corporation and Ken Schwaber-Scrum Alliance*, 1-28.
15. Lewis, J., & Neher, K. (2007). Over the Waterfall in a Barrel – MSIT Adventures in Scrum. *AGILE, IEEE Computer Society,* 389-394.
16. Luz, M., Gazineu, D., & Teófilo. M. (2009). Challenges on Adopting Scrum for Distributed Teams in Home Office Environments. *World Academy of Science, Engineering & Technology, 35*, 308-311.

17. Mahnič, V. (2011). A Case Study on Agile Estimating and Planning using Scrum. *ELEKTRONIKA IR ELEKTROTECHNIKA, 5*(111), 123-128.
18. Mahnič, V., Georgiev, S., & Jarc, T. (2009). Poučevanje metode Scrum v sodelovanju s podjetjem za razvoj programske opreme. *Mednarodna multikonferenca Informacijska družba*, 12(A), 243-255.
19. Mahnič, V., & Hovelja, T. (2012). On using planning poker for estimating user stories. *The Journal of Systems and Software, 9*(85), 2086-2095.
20. Mahnič, V., & Žabkar, N. (2012). Measuring Progress of Scrum-based Software Projects. *ELEKTRONIKA IR ELEKTROTECHNIKA, 8*(18), 73-76.
21. Marchenko, A., & Abrahamsson, P. (2008). Scrum in a Multiproject Environment: An Ethnographically-Inspired Case Study on the Adoption Challenges. *IEEE Computer Society*, 15-26.
22. Misra, S., Kumar, V., Kumar, U., Fantazy, K., & Akhter, M. (2010). Agile software development practices: evolution, principles, and criticisms. *International Journal of Quality & Reliability Management, 9*(29), 972-980.
23. Olson, K. (2010). HEI Embraces Scrum Project Management. *SMT Magazine*, 5(25), 44-52.
24. Pichler, R. (2010). *Agile Product Management with Scrum* (1th ed.). Boston: Addison-Wesley Professional.
25. Pries, K., & Quigley, J. (2010). *Scrum Project Management* (1th ed.). Boca Raton: CRC Press.
26. Qumer, A., & Henderson-Sellers, B. (2008). An evaluation of the degree of agility in six agile methods and its applicability for method engineering. *Information and Software Technology, 4*(50), 280–295.
27. Salo, O., & Abrahamsson, P. (2008). Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum. *The Institution of Engineering and Technology, 1*(2), 58-64.
28. Schatz, B., & Abdelshafi, I. (2005). Primavera Gets Agile: A Successful Transition to Agile Development. *IEEE Computer Society*, 3(22), 36-42.
29. Schiel, J. (2009). *Enterprise-Scale Agile Software Developmen* (1th ed.). Boca Raton: CRC Press.
30. Schwaber, K. (1995). SCRUM Development Process. *Proceedings of the Conference on Object-Oriented Programing Systems, Languages, and Applications Workshop on Business Object Design and Implementation*, 1995, 117-134.
31. Schwaber, K. (2004). *Agile Project Management with Scrum* (1th ed.). Washington: Microsoft Press.
32. Schwaber, K., & Sutherland, J. (2011, oktober). Scrum vodič. *Scrum.org*. Najdeno 10. Aprila 2013 na spletnem naslovu https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum%20Guide%20-%20SI.pdf#zoom=100.
33. Sutherland, J. (2010). *Scrum Handbook*. Boston: Scrum Training Institute.
34. Urevc, J., & Mahnič, V. (2012). Ocena prednosti metode Scrum in njenih tipičnih praks. *Uporabna informatika, 10*(3), 184-194.

35. Vinekar, V., & Slinkman, C., Nerur, S. (2006). Can Agile and Traditional Systems Development Approaches Coexist? An Ambidextrous View. *Information Systems Management, 3*(23), 3, 31-42.
36. Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011). The agile requirements refinery: Applying SCRUM principles to software product management. *Information and Software Technology, 1*(53), 58-70.
37. Williams, L. (2010). Agile Software Development Methodologies and Practices. *Advances In Computers, 1*(80), 1-44.
38. Woodward, E., Surdek, & S., Ganis, M. (2010). *A Practical Guide to Distributed Scrum* (1th ed.). Boston: IBM Press.