UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

MAGISTRSKO DELO

UPORABA AGILNE METODOLOGIJE SCRUM PRI PROJEKTIH RAZVOJA PROGRAMSKE OPREME

Ljubljana, oktober 2021 CLAUDIA MANAGURE

IZJAVA O AVTORSTVU

Podpisana Claudia Managure, študentka Ekonomske fakultete Univerze v Ljubljani, avtorica predloženega dela z naslovom Uporaba agilne metodologije Scrum pri projektih razvoja programske opreme, pripravljenega v sodelovanju s svetovalcem red. prof. dr. Talib Damij

I Z J A V L J A M

1. da sem predloženo delo pripravila samostojno;

2. da je tiskana oblika predloženega dela istovetna njegovi elektronski obliki;

3. da je besedilo predloženega dela jezikovno korektno in tehnično pripravljeno v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani, kar pomeni, da sem poskrbela, da so dela in mnenja drugih avtorjev oziroma avtoric, ki jih uporabljam oziroma navajam v besedilu, citirana oziroma povzeta v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani;

4. da se zavedam, da je plagiatorstvo – predstavljanje tujih del (v pisni ali grafični obliki) kot mojih lastnih – kaznivo po Kazenskem zakoniku Republike Slovenije;

5. da se zavedam posledic, ki bi jih na osnovi predloženega dela dokazano plagiatorstvo lahko predstavljalo za moj status na Ekonomski fakulteti Univerze v Ljubljani v skladu z relevantnim pravilnikom;

6. da sem pridobila vsa potrebna dovoljenja za uporabo podatkov in avtorskih del v predloženem delu in jih v njem jasno označila;

7. da sem pri pripravi predloženega dela ravnala v skladu z etičnimi načeli in, kjer je to potrebno, za raziskavo pridobila soglasje etične komisije;

8. da soglašam, da se elektronska oblika predloženega dela uporabi za preverjanje podobnosti vsebine z drugimi deli s programsko opremo za preverjanje podobnosti vsebine, ki je povezana s študijskim informacijskim sistemom članice;

9. da na Univerzo v Ljubljani neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve predloženega dela v elektronski obliki, pravico reproduciranja ter pravico dajanja predloženega dela na voljo javnosti na svetovnem spletu preko Repozitorija Univerze v Ljubljani;

10. da hkrati z objavo predloženega dela dovoljujem objavo svojih osebnih podatkov, ki so navedeni v njem in v tej izjavi.

V Ljubljani, dne 7.10.2021 Podpis študentke: _____

KAZALO

KAZALO SLIK

KAZALO TABEL

SEZNAM KRATIC

angl. – angleško
CRM – (angl. Customer Relationship Management); Sistem za upravljanje odnosov s strankami
CRP – (angl. Conference Room Pilot); Metoda predstavitve pilotne rešitve
ERP – (angl. Enterprise Resourse Planning); Celovita programska rešitev
FDD – (angl. Feature driven Development); Funkcijsko voden razvoj
IS – informacijski sistem
IT-Infrastruktura – (angl. Information Technology Infrastructure); Informacijsko tehnološka infrastruktura
XP – (angl. Extreme Programming); ekstremno programiranje

UVOD

Področje razvoja programske opreme se srečuje z naslednjimi težavami: proces razvoja je časovno predolg, stroški so previsoki in programska oprema, ko je enkrat dokončno razvita, ne deluje po pričakovanjih (Fitzgerald, Hartnett & Conboy, 2006). Razlogov za to je več. Med njimi tudi samo poslovno okolje, ki bolj kot kadarkoli prej zahteva fleksibilnost. Kot sta že leta 1986 rekla Takeuchi in Nonaka (1986), je za uspeh na trgu k visoki kvaliteti, nizkim stroškom in diferenciaciji potrebno dodati še hitrost in prilagodljivost.

Nepredvidljivost okolja in posledične spremembe so stalnica. Planiranje vnaprej pri vse večji kompleksnosti ne daje več želene stabilnosti. Agilne metode kot odgovor temu stanju zagovarjajo sprejemanje sprememb in jih smatrajo prej kot priložnosti za izboljšanje produkta kot pa grožnjo. Fowler in Highsmith (2001) spremembe primerjata s povratnimi informacijami, ki jih razumemo kot nekaj pozitivnega, vendar prav tako vodijo v spremembe.

Prav tako rigidne metode ali pomanjkanje metod nasploh vplivajo na težave v organizaciji, pri upravljanju ljudi in v kvaliteti razvitega produkta. Fitzgerald, Hartnett in Conboy (2006) pravijo, da empirične raziskave kažejo na precej omejeno uporabo metod med razvijalci. Prav tako birokratska narava rigidnih metod upočasnjuje razvoj in povzroča t. i. izgubo cilja; to se zgodi, ko se razvijalci toliko ukvarjajo s samo metodo, da pozabijo na končni cilj, tj. razvoj uporabne programske opreme.

Pri projektnem managementu pogosto razlikujemo med tradicionalnimi plansko usmerjenimi metodami in kasnejšimi agilnimi metodami. Poznan primer planske je slapovna metoda, ki obsega obsežno raziskovanje in planiranje zahtev na začetku, tekom razvoja pa se giblje kot slap iz ene v drugo fazo, brez možnosti vrnitve na predhodno. Kljub temu, da ima projektni management kot disciplina korenine v planskih metodah, priljubljenost uporabe agilnih metod med projektnimi managerji narašča.

Plansko usmerjene metode so bile razvite v drugačnem času, kot je danes. Wysocki (2009) sodobno okolje, v katerem se razvija programska oprema, označuje z visoko hitrostjo, pogostimi spremembami, nizkimi stroški, kompleksnostjo in negotovostjo. Pogoste tehnološke inovacije so stalnica in digitalna transformacija je za podjetja nujna za nadaljnje uspešno poslovanje. Statistike kažejo, da je bilo v zadnjih dveh letih ustvarjeno 90 % masovnih podatkov (angl. big data), da bo do leta 2030 globalno število pametnih naprav naraslo na 50 bilijonov in da globalno trenutno deluje 1,35 milijona tehnološko usmerjenih zagonskih podjetij (angl. start up) (Bulao, 2021). V nasprotju z danes, je bila včasih življenjska doba programske opreme dovolj dolga, da je povrnila stroške tako dolgotrajnega razvoja. Zaradi večje konkurence in vse hitrejšega tehnološkega razvoja se je življenjska doba uporabe programske opreme občutno zmanjšala in tako ne opravičuje več večletnega razvoja. Novejše agilne metode predvidevajo hiter razvoj osnovne verzije in nadaljnjo

inkrementalno dobavo novih izdaj programske opreme. Zaradi hitrejše razpoložljivosti funkcionalne verzije je to možno hitreje lansirati na trg ter že tekom razvoja pridobivati prihodke in pričakovati hitrejšo povrnitev investicije (Quigley & Pries, 2011).

Od tradicionalnih planskih metod se agilne metode razlikujejo po pristopu do sprememb, pristopu do komunikacije z deležniki, planiranju in učenju tekom razvoja. Po agilnih metodah se spremembe tekom razvoja sprejemajo − spremembe zahtev tekom razvoja se smatrajo kot pot do bolj kvalitetnega produkta in ne kot problem. Prav tako so nove informacije ali spremembe iz okolja tekom življenjskega cikla razvoja nekaj pričakovanega; le-teh ni mogoče v zadostni meri predvideti na začetku življenjskega cikla razvoja; zato je bolj smiselno sredstva, namenjena obsežnemu vnaprejšnjemu planiranju/predvidevanju sprememb iz okolja, preusmeriti drugam.

Plansko usmerjene metode gradijo na predpostavki, da so vse potrebne informacije za planiranje razvoja na voljo na začetku procesa in je s tem mogoče ustvariti natančen plan. Pri agilnih metodah se prav to razume drugače; noben od deležnikov na začetku procesa ne more imeti popolnega pregleda in vseh informacij. Zahteve se tekom razvoja produkta spreminjajo, prav tako prioritete. Tekom projekta se pridobijo nova znanja, pomembni izsledki in jasnejša vizija produkta. Zato je tekom razvoja potrebna fleksibilnost, ki omogoča spremembe zahtev in posledično razvoj ustreznih funkcionalnosti (Koch, 2005).

Med drugim so bile kot odgovor na pomanjkanje fleksibilnosti in kredibilnosti plansko usmerjenih metod razvite agilne metode, med njimi tudi Scrum, ki obljublja nižje stroške, boljšo produktivnost tima, višje zadovoljstvo naročnika in boljšo kvaliteto končnega produkta. Rubin in Lichtenberg (2014) jo opredelita kot enostavno, ljudem prilagojeno metodo, ki temelji na vrednotah iskrenosti, odprtosti, pogumu, spoštovanju, osredotočenosti, zaupanju, vzajemni krepitvi in sodelovanju.

O prednostih Scrum metode poročajo tudi Fitzgerald, Hartnett in Conboy (2006), ki pri podjetju Intel Shannon zabeležijo kar nekaj pozitivnih učinkov uvedbe metode. Planiranje in sledenje preko dnevnih sestankov in uporabe skupne table za pregled nalog postane skupna naloga celotnega tima. Prav tako je bila komunikacija in morala znotraj timov odlična. Kljub kompleksnim projektom so timi dosegali zadane roke z vzdrževanjem visokega nivoja kvalitete produkta.

V magistrski nalogi želim preučiti uporabo metode Scrum za povečanje uspešnosti pri projektih razvoja programske opreme ter način uvedbe metode na primeru podjetja. Zanima me, kakšna je uporaba agilne metode Scrum v praksi in ali je podjetje z uvedbo metode povečalo učinkovitost projektnega managementa in uspešnost projektov razvoja programske opreme. Sprašujem se ali so projekti izpeljani hitreje in z nižjimi stroški ter ali je produkt bolj kvaliteten, kar vodi k večjemu zadovoljstvu naročnikov.

Nato se sprašujem ali je za učinkovito uporabo metode Scrum potrebno metodo prilagoditi konkretnim potrebam podjetja v praksi. Vprašanje se nanaša na uvedbo metode v

preučevanem oddelku. Alternativa uvedbi metode v celoti je prilagoditev določenih praks ali uvedba metode zgolj pri nekaterih procesih ali fazah. Kljub temu, da je metodo Scrum razumeti kot skupek smernic, ni nujno, da je primerna za vsa podjetja in projekte.

Cilj raziskave je potrditev ali zavrnitev koristi, ki naj bi jih po teoriji prinesla uporaba metode Scrum ter boljše razumevanje te metode v praksi. Nadalje želim raziskati način uvedbe metode, in sicer želim izvedeti:

- kakšen je bil potek uvedbe,
- kako se je razdeljevalo vloge, predvidene po metodi Scrum,
- s kakšnimi ovirami se je podjetje srečevalo in kje so se pojavile težave pri uvedbi, katere težave so bile predvidene in katere nepredvidene,
- v kolikšni meri so se odločili za prilagoditev metode Scrum lastnim potrebam,
- je bila metoda Scrum uvedena v celoti ali v kombinaciji z drugimi metodami in zakaj.

Preko pregleda strokovne literature najprej prikažem agilne metode in metodo Scrum v teoriji. Za pridobitev vpogleda v projektni management po metodi Scrum v praksi se odločim za kvalitativno metodo raziskovanja: polstrukturirani intervju. Pristop je poizvedovalne narave, saj sem želela ne samo odgovoriti na zadana raziskovalna vprašanja, ampak zajeti morebitne dodatne nepričakovane informacije, povezane z uporabo metode Scrum v praksi. Intervju sem opravila s ključno osebo v procesu, odgovorno za uvedbo metode Scrum v oddelek.

Magistrska naloga je sestavljena iz štirih poglavij. V prvem poglavju obravnavam teorijo projektnega managementa in opredeljujem osnovne koncepte, kot so železni trikotnik, življenjski cikel projekta in razvoja produkta. Poglobim se tudi v definicijo uspešnosti projekta; kako prepoznamo uspešno izveden projekt in kateri faktorji vplivajo na verjetnost uspešnega projekta. Drugo in tretje poglavje sta namenjeni agilnim metodam, kjer se še posebej poglobim v podrobnosti metode Scrum. V empiričnem delu prikažem rezultate poglobljenega intervjuja. Prikažem uvedbo metode Scrum v oddelek in orišem management projektov z metodo.

1 PROJEKTNI MANAGEMENT

1.1 Opredelitev projekta

Project Management Institute (2004) definira projekt kot začasno prizadevanje z namenom ustvarjanja edinstvenega produkta, storitve ali rezultata. Vsak projekt ima definiran začetek in konec. Projekt se konča, ko je cilj dosežen ali pa je jasno, da cilja ni več mogoče doseči in je projekt ustavljen. Časovna omejenost projekta pa ne pomeni, da je projekt nujno kratkotrajen, saj lahko projekt traja tudi več let. Projekti so možni na vseh nivojih podjetja, vpletenih je lahko poljubno število udeležencev, razlikuje pa se tudi struktura znotraj

projektnega tima. Značilno za projektno delo je postopno približevanje cilju. Medtem ko je na začetku obseg definiran zgolj grobo, se tekom razvoja specifike obsega natančneje določa in cilj postaja bolj konkreten.

Edinstvenost rezultata in začasnost pa projekt razlikujeta od ostalih aktivnosti v podjetju – torej ponavljajočih se operativnih procesov v podjetju. Prav tako ima projekt običajno za cilj neko izboljšavo, novost, rešitev nekega problema, medtem ko operativni procesi običajno skrbijo za izvajanje osnovnih aktivnosti podjetja, so ponavljajoči in vedno v teku. Brez operativnih del podjetje kratkoročno ne bi preživelo, brez projektnega dela pa verjetno ne dolgoročno. To ne pomeni, da so projekti pomembnejši ali težavnejši kot operativni procesi, le da se potrebe in pristopi k aktivnostim zaradi različnih ciljev razlikujejo (Project Management Institute, 2004).

Projekti so pogosto povezani s strateškim načrtovanjem podjetja in namenjeni doseganju ciljev. Project Management Institute (2004) navaja naslednje pogoste pobudnike projektov: odziv na tržno povpraševanje, potreba podjetja (npr. razvoj novega produkta za povečanje prihodkov), povpraševanje naročnika, pridobivanje tehnološke prednosti, zadoščanje pravnim zahtevam. Če podjetje želi pridobiti in obdržati konkurenčno prednost, so projekti neizogibni, projekti imajo cilje, ki podjetje potiskajo naprej.

Projekt je lahko samostojen ali pa je del programa. Program je skupina med seboj povezanih projektov, ki se jih bolj učinkovito upravlja skupaj kot posamično. Program management je centralizirano in koordinirano upravljanje skupine projektov za doseganje zastavljenih strateških ciljev programa. Programi pa so lahko zopet del portfelja projektov podjetja, ki omogoča učinkovito upravljanje za doseganje strateških poslovnih ciljev podjetja (Project Management Institute, 2004).

Med informacijsko tehnološke projekte pa klasificiramo tudi projekte razvoja programske opreme, ki jih Wysocki (2006) definira kot kompleksno prizadevanje dveh ali več oseb, ki znotraj časovnih in stroškovnih omejitev ter omejitev človeških virov ustvarijo novo ali izboljšano računalniško kodo, ki doprinaša poslovno vrednost novemu ali obstoječemu poslovnemu procesu. Chemuturi (2013) definira projekt na področju informacijske tehnologije kot projekt, ki vzpostavi informacijsko tehnološko infrastrukturo (angl. Information Technology Infrastructure, v nadaljevanju IT-infrastruktura) v podjetju in vključuje vse lokacije in oddelke vpletene v izbrani postopek procesiranja informacij v podjetju. Avtor navaja naslednje tipe projektov na področju informacijske tehnologije:

- implementacija nove IT-infrastrukture,
- dodelava obstoječe IT-infrastrukture (npr. pri povečanju informacijskih potreb),
- nadgradnja in posodobitev obstoječe IT-infrastrukture,
- migracija obstoječe IT-infrastrukture,
- vzdrževanje obstoječe IT-infrastrukture.

1.2 Opredelitev projektnega managementa

Projektni management kot samostojna disciplina se je razvil med drugo svetovno vojno, ko so Američani prvič s pomočjo tehnik projektnega managementa razvijali orožja. Ti projekti so bili tako obsežni, tudi finančno, da jih ni bilo več mogoče upravljati z obstoječimi metodami managementa. Takrat sta se razvili še danes znani tehnika upravljanja in pregledovanja programa (projekta) (angl. Program Evaluation and Review Technique) in metoda kritične poti (angl. critical path method). Začetek projektnega managementa kot poklica lahko vidimo ob ustanovitvi Združenja projektnega managementa (angl. International Project Management Association) leta 1965 in Inštituta projektnega managementa (angl. Project Management Institute) leta 1969. Združenji od takrat naprej skrbita za profesionalizacijo poklica, vzpostavitev standardov in omogočata certificiranje znanja projektnega managementa (Project Management Institute, 2004).

Project Management Institute (2004) projektni management definira kot aplikacijo znanja, spretnosti, orodij in tehnik projektnih aktivnostih z namenom doseganja projektnih zahtev. S tem neposredno povezan cilj je zadovoljevanje ali preseganje potreb in pričakovanj deležnikov. Deležniki so posamezniki in organizacije, ki so vpleteni v projekt ali imajo s projektom povezan interes. Zato je v sklopu projektnega managementa pomembno, da se deležnike, njihove potrebe in vpliv spozna že na začetku projekta. Deležniki imajo vpliv na zahteve in tako je razumevanje in upravljanje pričakovanj deležnikov ključno za uspeh projekta. Kot pravi Verzuh (2011) je upravljanje pričakovanj ena od glavnih nalog projektnih managerjev.

Poleg poznavanja procesov in tehnik projektnega managementa pa mora projektni manager poznati tudi druga področja za uspešno delo. Project Management Institute (2004) poleg znanja projektnega managementa navaja še naslednja potrebna področja znanja:

- znanja, standardi in predpisi strokovnega področja projekta,
- razumevanje okolja projekta, kar vključuje družbeni, kulturni, ekonomski, politični in fizični kontekst,
- osnovno znanje managementa,
- medosebne veščine, ki so bistvene za uspešno komunikacijo in motivacijo, vplivanje in prepričevanje ter predvsem preprečevanje konfliktov in reševanje problemov.

Projektni management je sicer od stroke neodvisna disciplina, ampak je za delo znotraj neke stroke potrebno vsaj grobo poznavanje le-te. Projektni management se npr. uporablja tako na področju gradbeništva, razvoja informacijskih sistemov kot tudi pri razvoju tržno komunikacijskih strategij, vendar je projektni manager lahko uspešen le z vsaj grobim poznavanjem panoge ali področja projekta. Ne rabi biti strokovnjak, vendar brez poznavanja področja ne more ustrezno voditi in motivirati strokovnega projektnega tima ter sprejemati kompetentnih odločitev. Skladno s tem Verzuh (2011) poudarja, da je projektni management

od strokovnega področja neodvisna disciplina, to pa ne velja za projektne managerje, ki za uspešen management projektov potrebujejo strokovno znanje s področja.

Vsak projekt je vpleten v večdimenzionalno okolje, katerega karakteristike imajo na projekt ali pozitiven ali negativen vpliv; v vsakem primeru pa je projekt odvisen od okolja, v katerem se izvaja. Projektni manager mora zato okolje dobro poznati (npr. družbeni in ekonomski kontekst sta pogosto pomembna) in ga upoštevati pri planiranju in managementu projekta. Za projektnega managerja je zaradi narave dela, ki zahteva čim bolj popolno sliko projekta in njegovo povezanost znotraj podjetja, pomembno, da pozna vse funkcije managementa v podjetju – od financ, preko prodaje in marketinga do osnov logistike in kadrovanja. Prav tako mora biti projektni manager dober komunikator. To ne pomeni, da mora zgolj učinkovito komunicirati, ampak pomeni, da mora znati tudi izkoriščati svoj vpliv v podjetju, imeti vodstvene sposobnosti in emocionalno inteligenco, s katero motivira tim, imeti pogajalske sposobnosti ter sposobnost učinkovitega reševanja problemov (Project Management Institute, 2004).

Projektni management razvoja programske opreme Wysocki (2006) definira kot disciplino ocenjevanja značilnosti potrebne programske opreme, izbora ustreznega življenjskega cikla razvoja in aplikacijo ustreznega pristopa projektnega managementa z namenom zadovoljevanja potreb naročnika po čimbolj učinkovitem doprinosu poslovnih vrednosti. Značilnosti programske opreme avtor razdeli v štiri kategorije, ki se razlikujejo po jasnosti cilja in jasnosti rešitve za doseganje cilja. Poudarja da iz teh značilnosti izhaja izbor primernega pristopa k projektu. Pri jasnem cilju in rešitvi je primernejši planski pristop, pri projektih z več neznankami pa je primernejši agilni pristop. Predvsem pa avtor poudarja spoštovanje časa kot pomembnega vira in izogibanje delu brez dodane vrednosti v smeri cilja projekta.

1.2.1 Železni trikotnik časa, stroškov in kvalitete

Projektni manager operira s tremi med seboj povezanimi spremenljivkami znotraj t. i. železnega trikotnika: čas, stroški in kvaliteta/obseg. Primer: projekt lahko zaključimo hitreje, če povečamo stroške ali znižamo kvaliteto; če znižamo stroške to lahko pomeni kasnejši zaključek projekta ali nižjo kvaliteto; če želimo višjo kvaliteto ali vključitev dodatnih zahtev, moramo računati s povečanimi stroški in/ali kasnejšim zaključkom projekta. Izziv je najti optimalno ravnovesje med spremenljivkami, pri čemer je običajno vsaj ena izmed njih fiksna.

Železni trikotnik stroškov, časa in kvalitete predstavlja pomembno ravnovesje, ki ga mora projektni manager na začetku vzpostaviti in ustrezno vzdrževati med projektom. Uspešnost projekta pa ne temelji zgolj na upravljanju ravnovesja, ampak tudi na komunikaciji z deležniki. Kot pravi Verzuh (2011), je dojemanje drugih tisto, kar določa uspešnost projekta in poudarja pomembnost komunikacije realistično definiranega trikotnika in morebitnih sprememb z vsemi deležniki.

1.2.2 Funkcije projektnega managementa

Management projektov vključuje tehnike, ki jih Verzuh (2011) povzame v tri funkcije, ki pa niso zaporedne, ampak jih projektni manager opravlja oz. ponavlja dnevno:

- Prva funkcija je definiranje/iniciacija (angl. initiation). Projektni manager je odgovoren za jasno definiranje ciljev in omejitev ter komuniciranje, usklajevanje in potrditev le-teh s strani deležnikov. Določiti mora osnovna pravila. Med drugim mora definirati člane projektnega tima, njihove vloge in definirati procese komunikacije.
- Pri planiranju projekta postane projektni manager konkretnejši in z orodji in tehnikami projektnega managementa ustvari plan implementacije. Sem spadajo tudi aktivnosti odkrivanja in preprečevanja tveganj.
- Pomembna funkcija pa je tudi kontrola projekta. Projektni manager tekom projekta z definiranimi kazalniki uspešnosti in planom spremlja napredek projekta in ga s korektivnimi aktivnostmi in dobro komunikacijo s timom usmerja k cilju.

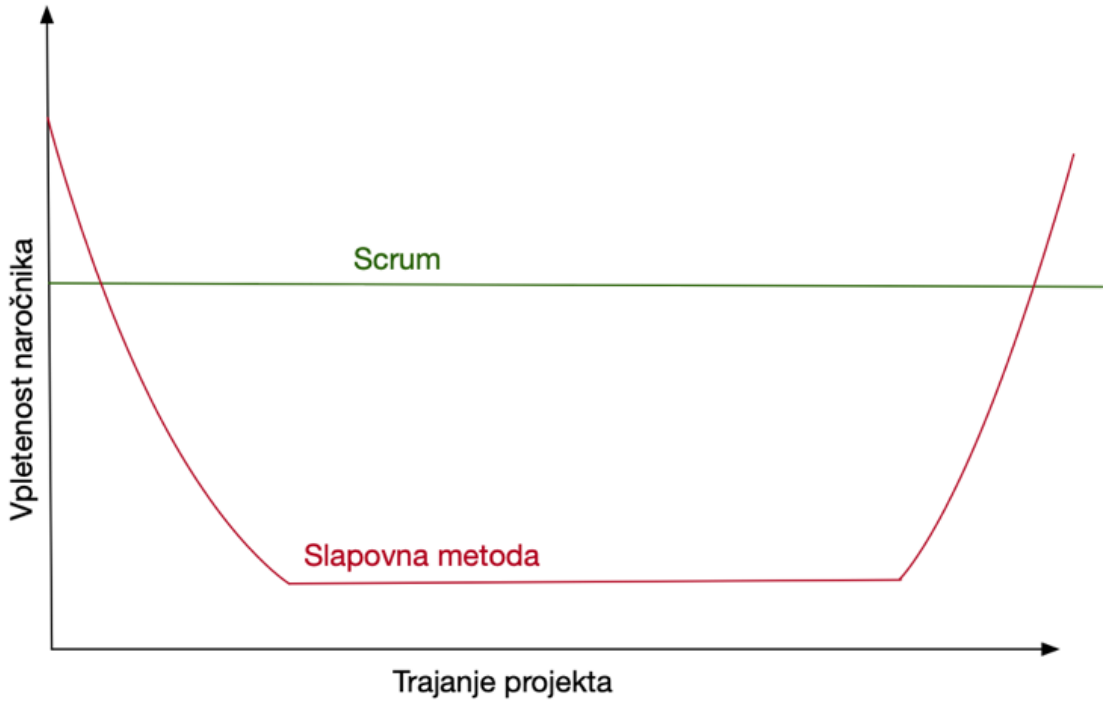
1.2.3 Razlike med planskim in agilnim projektnim managementom

Planski princip managementa projektov temelji na zaporednem sosledju korakov, kjer se, kolikor je to le mogoče, ne vrača na predhodni korak. Planiranje je obsežno in obsega projekt celostno, od uvodnega sestanka (angl. kick-off meeting) do predaje programske rešitve naročniku. Na začetku projekta v fazi planiranja se izvaja obsežne analize, na podlagi katerih se zapišejo zahteve in ustvari plan. Roki so fiksni in spremembe niso zaželene. Cilj je doseči stabilnost in predvidljivost.

Kot pri planskih metodah je tudi pri agilnih potrebno planiranje. Razlika je zgolj v tem, da planiranje ni tako v ospredju in ni osredotočeno na začetek projekta, ampak se planiranje izvaja skozi celoten projekt in to ravno toliko, kot je potrebno in ravno takrat, ko je to potrebno. Izvajalci metode Scrum verjamejo, da preveč vnaprej definiran plan ne more biti pravilen. Rubin in Lichtenberg (2014) poudarjata, da metoda Scrum temelji na prilagajanju planov, saj jih ne moremo vnaprej pripraviti z zadostno mero natančnosti. Planiranje vnaprej označita kot »zapravljanje časa«, saj na začetku nimamo vseh informacij in je smiselno odločitve ali planiranje opraviti kasneje, ko so nam te informacije na voljo.

Pomembna razlika je vpletenost naročnika. Na sliki 1 vidimo kako je pri planskih metodah naročnik vpleten zgolj v začetni analitično-planski fazi in ob zaključku projekta. S tem planske metode vsebujejo večje tveganje, da produkt ne bo ustrezal potrebam naročnika. Tveganje pri agilnih metodah zmanjšujeta iterativna in inkrementalna narava procesa, kjer je skozi iteracije s postopnim dodajanjem funkcionalnosti produkt vedno znova pregledan in predstavljen naročniku. S tem je verjetnost, da bo produkt ustrezal potrebam naročnika, večja (Rubin & Lichtenberg, 2014).

Slika 1: Vpletenost naročnika glede na čas pri slapovni metodi in metodi Scrum



Prirejeno po Rubin & Lichtenberg (2014).

Cobb (2015) navaja pet ključnih prednosti agilnega pristopa managementa projektov:

- Agilni pristop se osredotoča na poslovne rezultate in poskuša z vsako iteracijo povečevati vrednost produkta in zadovoljiti poslovne potrebe naročnika. Z vključenostjo uporabnika v razvoj in konstantno komunikacijo je bolj verjetno, da bo produkt imel želeno vrednost.
- Produkt je hitreje pripravljen za trg. V nasprotju s planskim pristopom je začetek implementacije projekta pri agilnem pristopu možen bistveno hitreje. Faza začetnega planiranja je bistveno krajša. Funkcionalnosti se dodajajo inkrementalno. To pomeni, da je produkt že zgodaj v razvoju funkcionalen. Prav tako pa se z osredotočanjem na preprostost ter izogibanje delu in funkcionalnostim z malo dodane vrednosti prispeva k večji učinkovitosti pri razvoju.
- Agilni pristop dosega višjo produktivnost in ima nižje stroške. Produktivnost tima je zaradi samoorganiziranosti in višje motiviranosti članov višja. Z boljšo komunikacijo in pogostim pregledom dela se izognemo nepotrebnim funkcionalnostim, ki ne prinašajo dodane vrednosti.
- Kvaliteta produkta je višja, saj testiranje produkta leži že v odgovornosti razvojnega tima in ne zgolj oddelka za upravljanje kvalitete.
- Beleži se višja organizacijska učinkovitost. V okolju s pozitivnimi vrednotami in medsebojnim spoštovanjem, so ljudje bolj motivirani za svoje delo. Agilni pristop spodbuja sodelovanje in strmenje k skupnim ciljem na vseh nivojih. To doprinaša k višji učinkovitosti podjetja.

Prav tako zasledimo nekaj slabosti agilnega pristopa. S sprejemanjem sprememb pozno v razvoju se pri agilnem pristopu odpove določeni stopnji kontrole in predvidljivosti. Prav tako so zahteve podrobneje definirane šele med projektom. To pomeni, da je tveganje napak v ocenah časa, stroškov in obsega toliko večje. Za nekatere projekte, za katere je natančnost ocen ključna za uspešnost, agilne metode niso primerne.

Za oceno se uporablja več metod, med katerimi kot najpogostejšo avtorji Ceschi, Sillitti, Succi in De Panfilis (2005) zabeležijo ocenjevanje na podlagi preteklih projektov in izkušenj. Vendar avtorji poudarjajo, da je težko imeti zadostno količino homogenih preteklih projektov za zanesljivo oceno, saj se projekti preveč razlikujejo. Tudi Quigley in Pries (2011) ugotavljata, da je v ocenah vedno prisotno ugibanje, tudi če se poskuša skozi razčlenitev na posamezne naloge podati nekoliko bolj realistično oceno. Pogosto pride do napak ocen pri planiranju in ocenjevanju zahtev, katerih implementacija sega daleč v prihodnost.

Cobb (2015) poudarja, da je potrebno najti ustrezno ravnotežje med planskim in agilnim pristopom glede na potrebe konkretnega projekta. Prav tako Wysocki (2006) izbor pristopa pogojuje z lastnostmi projekta na oseh jasnosti cilja in rešitve. Manj kot sta jasna oziroma več kot je pričakovanih sprememb in negotovosti v okolju, bolj primeren je agilni pristop.

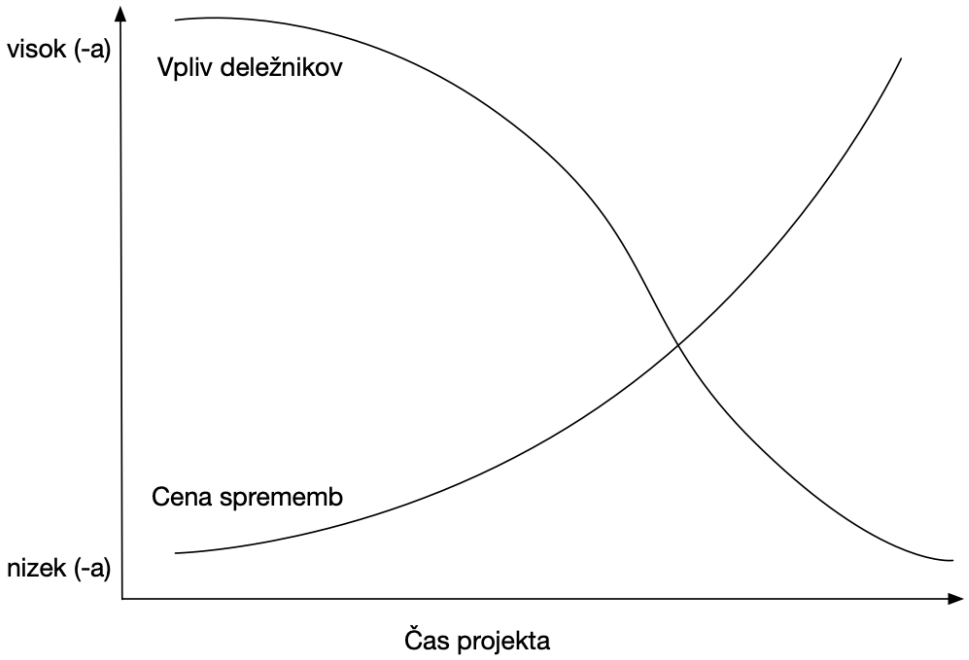
1.3 Življenjski cikel projekta

Projekt poteka skozi zaporedne faze, ki jih imenujemo življenjski cikel projekta. Namen je omogočanje ustrezne kontrole nad projektom, ki omogoča usmerjanje k definiranemu cilju oziroma željenemu produktu, storitvi ali rezultatu.

Strukture življenjskih ciklov projektov se razlikujejo. To ni presenetljivo glede na veliko raznolikost projektov, ki se razlikujejo po velikosti, kompleksnosti, trajanju, stroki, produktu, riziku, okolju itd. Konec vsake faze običajno označuje nek mejnik, povezan z odločitvami. Pogosto je ob koncu faze predstavljen rezultat dela, ki se ga pregleda in v pozitivnem primeru potrdi kot začetek naslednje faze. Faze potekajo zaporedno, ampak je možno tudi prekrivanje in iterativno prehajanje faz, npr. kadar je po fazi planiranja potrebna sprememba osnovnih definicij iz predhodne faze (Project Management Institute, 2004; Verzuh, 2011).

Stroški so najvišji na sredini projekta, saj je tam tudi najbolj koncentrirano delo. Ko projekt prehaja končne faze, se običajno zmanjšuje možnost vpliva na končni produkt; spremembe proti koncu življenjskega cikla so problematične. Slika 2 prikazuje, kako se vpliv deležnikov s časom zmanjšuje, prav tako pa narašča cena sprememb (Project Management Institute, 2004).

Slika 2: Vpliv deležnikov in cena sprememb tekom projekta



Prirejeno po Project Management Institute (2004).

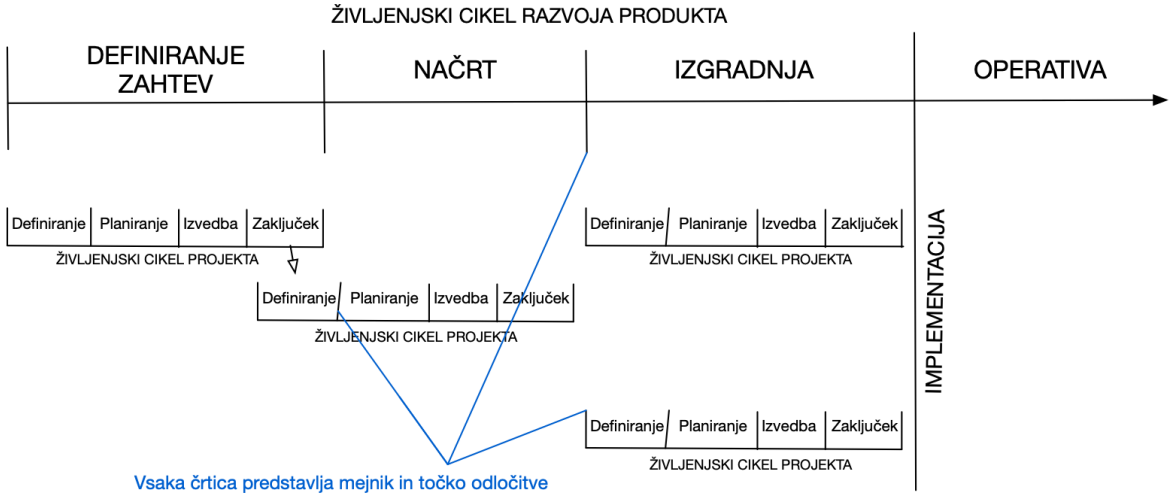
Število faz v ciklu je lahko različno, kot standardne faze pa Verzuh (2011) navaja štiri:

- Definiranje/iniciacija, kjer se definirajo cilji, grob pristop in omejitve projektnega trikotnika. Definirajo in zapišejo se pravila, ki jih potrdijo vsi vpleteni. V tej fazi se zapiše listino projekta (angl. project charter) in opredeli obseg (angl. scope statement). To fazo Project Management Institute (2004) označi s pojmom iniciacija.
- Planiranje, katerega rezultat je podrobnejši načrt pristopa k razvoju, je druga faza.
- Izvedba je najintenzivnejša faza, v kateri se razvije ciljni produkt ali rezultat.
- Na koncu je zaključek, kjer se projekt formalno zaključi ter preda naročniku ali v naslednjo fazo (npr. operativni proces ali nadaljnji razvoj). Zelo pomemben del je retrospektiva, pregled dela in ustvarjanje znanja za bodoče projekte.

1.4 Življenjski cikel razvoja produkta

Življenjski cikel projekta se ne sme enačiti ali zamenjati z življenjskim ciklom razvoja produkta. Projektni življenjski cikel je običajno vključen v del produktnega življenjskega cikla, npr. je razvoj spletne aplikacije zgolj eden izmed projektov digitalizacije produkta. Verzuh (2011) produktni življenjski cikel definira skozi naslednje faze: zahteve, načrtovanje, izgradnja in operativa ter poudarja, kot je prikazano na sliki 3, da lahko znotraj vsake faze najdemo življenjski cikel vsaj enega projekta. Produktni življenjski cikel opisuje potrebno delo za razvoj produkta ali programske opreme, medtem ko se projektni življenjski cikel ukvarja z managementom dela. Projekti so povezani z operativnimi procesi v podjetju in običajno ob zaključku sprožijo operativno uporabo rezultata projekta.

Slika 3: Življenjski cikel razvoja produkta



Prirejeno po Verzuh (2011).

Razvoj informacijskih sistemov se je pri uporabi linearnega, sekvenčnega modela razvojnega procesa pogosto soočal z neuspešnimi projekti, ki so vključevali produkte oziroma programsko opremo, ki ni dosegala želene vrednosti in ni zadoščala potrebam uporabnikov (Freedman, 2009). Pogosto je bil vzrok v zahtevah, ki jih je bilo težko zajeti, saj sami uporabniki niso imeli dovolj konkretne predstave o rešitvi, mogoče se celo niso zavedali, kaj točno potrebujejo. Tako se pri razvoju informacijskih sistemov uveljavlja iterativni in inkrementalni proces razvoja, ki predvideva hiter razvoj in dobavo manjšega dela programske rešitve z namenom čim hitrejše pridobitve povratne informacije uporabnikov za nadaljnji razvoj.

1.5 Uspešnost projekta

Podjetja želijo čim več projektov zaključiti uspešno. To je tudi razlog, zakaj mnogo podjetij prehaja k metodi Scrum managementa projektov. Standish Group International, Inc.(2015) med leti 2011 in 2015 beleži neuspeh pri zgolj 9 % projektov, vodenih po agilni metodi, in kar 29 % projektov, vodenih po slapovni metodi. Kot je razvidno iz tabele 1 je odstotek uspešnih projektov pri agilnih metodah skoraj štirikrat višji kot pri slapovni metodi.

Tabela 1: Chaos študija uspešnosti projekta glede na agilno ali slapovno metodo

Metoda projekta	Projekt uspešen	Projekt delno uspešen	Projekt neuspešen
Agilna metoda	39 %	52 %	9 %
Slapovna metoda	11 %	60 %	29 %

Prirejeno po Standish Group International, Inc. (2015).

Obstaja pa več definicij uspešnosti projekta. Možno je, da na prvi pogled uspešen projekt pri nekaterih deležnikih vendarle ni smatran kot uspešen. Npr. projekt razvoja programske opreme je iz vidika podjetja, ki ga je razvilo, uspešen, ker vanj ni bilo potrebno vložiti dodatnih sredstev, uporabnikom na drugi strani pa programska rešitev ne prinaša zadostne dodane vrednosti, ker je morebiti prišlo do neskladja med potrebami uporabnikov in definiranimi zahtevami. To predpostavlja večplastno definicijo uspeha, ki presega zgolj doseganje roka in omejitev proračuna.

V kontekstu projektnega managementa je to pomembno, saj se kvaliteto projektnega managementa povezuje s kvaliteto implementiranega informacijskega sistema in celostne uspešnosti projekta. Gollner in Baumane-Vitolina (2016) potrdita, da ima prav dimenzija kvalitete projektnega managementa pri implementaciji projektov celovitih programskih rešitev (angl. Enterprise Resource Planning, v nadaljevanju ERP) večjo težo na uspešnost projekta proti ostalim dimenzijam.

Standish Group International, Inc. (2015) prav tako spremeni svoje kriterije ocenjevanja uspešnosti projekta, tako da vključujejo tako uspešnost procesa projektnega managementa kot uspešnost končnega rezultata projekta. Svojim standardnim kriterijem projektnega trikotnika čas, proračun in cilj (angl. OnTime, OnBudget, OnTarget) dodajo metrike, ki merijo zadovoljstvo nad rezultatom projekta in s tem omogočijo, da so kot uspešni definirani tudi projekti, katerih rezultat je kljub nedoseganju zadanega obsega zadovoljiv s strani naročnika ali uporabnikov.

Ko govorimo o raziskovanju uspešnosti projekta, moramo ločevati med kriteriji, po katerih lahko po zaključku prepoznamo uspešno zaključen projekt (odvisno spremenljivko), in pa faktorji, ki pozitivno ali negativno vplivajo na verjetnost, da bo projekt uspešno izveden in zaključen (neodvisnimi spremenljivkami). Meritev uspešnosti projekta je ključna, če želimo oceniti vrednost ali učinkovitost naših projektnih aktivnosti. S poznavanjem in upoštevanjem faktorjev, ki vplivajo na uspešnost pa lahko izboljšamo možnosti uspešno zaključenega projekta.

1.5.1 Kriteriji uspešno zaključenega projekta

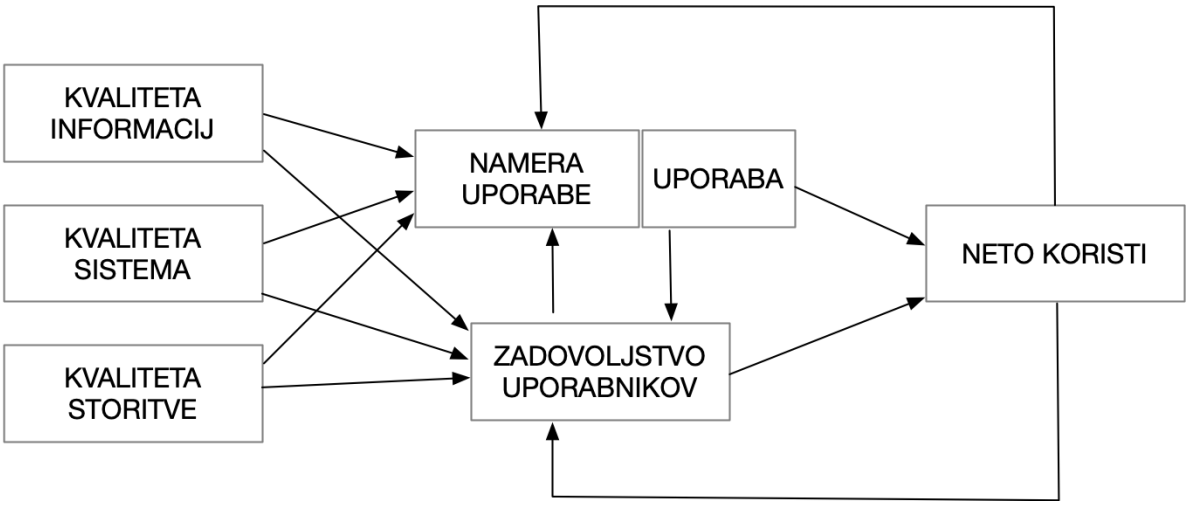
Pri projektnem managementu je potrebna učinkovita meritev uspešnosti projekta. Potrebno je definirati kriterije, po katerih je projekt smatran kot uspešen. Definicija uspešno zaključenega projekta, torej odvisne spremenljivke, pa ni najlažja. Najbolj razširjena definicija se nanaša na t. i. železni trikotnik (angl. iron triangle/triple constraint), ki uspešnost projekta definira na podlagi doseganja ciljev povezanih s časovnim in stroškovnim okvirom ter kvaliteto/obsegom.

Kriterij uspešnega projekta po tej definiciji je produkt, ki vključuje pogodbeno določene funkcionalnosti, ki so bile izvedene v roku in v okviru zastavljenega proračuna. Pomanjkljivost te splošne definicije je, da ne upošteva drugih kriterijev kot npr. zadovoljstva

naročnika. Projekta, pri katerem naročnik s programsko rešitvijo ni zadovoljen, tudi če je ta znotraj stroškovnih in časovnih okvirov ter vključuje definiran obseg zahtev, ne moremo smatrati kot uspešnega. Prav tako je možno, da se projekt zaključi s programsko rešitvijo, katere implementacija je presegla proračun ali pa je bila zaključena z zamudo, ampak rešuje poslovni problem in doprinaša k uspešnosti podjetja.

DeLone in McLean (2003) sta leta 1992 razvila večdimenzionalen model za meritev uspešnosti informacijskega sistema (v nadaljevanju IS). Kot prikazuje slika 4, vsebuje šest med seboj povezanih dimenzij: kvaliteta sistema, kvaliteta informacij iz sistema, uporaba informacijskega sistema, zadovoljstvo uporabnikov ter vpliv na posameznike in podjetje. Kasneje sta model predelala in k dimenzijam kvalitete dodala še kvaliteto storitve/podpore. Dimenzijo »uporaba« sta definirala natančneje kot namero uporabe in učinke na posameznike in podjetje združila v eno dimenzijo neto koristi, ki natančneje določa samo pozitivne koristi in ne tudi negativnih vplivov kot prej.

Slika 4: Posodobljeni Delone in McLean model uspešnosti



Prirejeno po Delone & McLean (2003).

Vsako spremenljivko uspešnosti DeLone in McLean (2003) definirata in podkrepita s primeri v tabeli 2. Kvaliteta sistema je definirana z želenimi lastnostmi sistema, npr. intuitivnost uporabe ali pa zanesljivost. S kvaliteto informacij označujemo kvaliteto podatkov in vsebin, ki jo pridobimo iz IS, npr. pomembno nam je, da so podatki točni in da jih dobimo ob pravem trenutku. Tehnična podpora, ki je hitro dosegljiva in je usposobljena za rešitev našega problema, je primer dobre kvalitete storitve. Uporaba sistema je povezana s spremenljivko zadovoljstva; način in količina uporabe vplivajo na naša stališča do IS in stopnje zadovoljstva. Neto koristi pa so razlog za uvedbo informacijskega sistema in jih avtorja definirata kot obseg, v katerem IS prispeva tako k uspešnosti posameznikov kot k uspešnosti podjetja, npr. povečanje produktivnosti ali povečanje prodaje.

Tabela 2: Spremenljivke uspešnosti informacijskega sistema

Spremenljivka uspešnosti IS	Definicija	Primeri meritve
Kvaliteta sistema	Želene lastnosti sistema.	Enostavnost in intuitivnost uporabe, zanesljivost, fleksibilnost IS.
Kvaliteta informacij	Želene lastnosti rezultatov IS (npr. poročila, preglednice).	Relevantnost, točnost, brezčasnost, uporabnost, popolnost informacij.
Kvaliteta storitve	Kvaliteta storitve ali podpore, ki je na voljo uporabnikom.	Odzivnost, zanesljivost, točnost, tehnična usposobljenost, empatija do uporabnikov.
Uporaba sistema	Stopnja in način, s pomočjo katerih uporabniki izkoriščajo sposobnosti IS.	Količina in frekvenca uporabe, ustreznost uporabe, namen uporabe.
Zadovoljstvo uporabnikov	Stopnja zadovoljstva uporabnikov z IS.	Lestvice za meritev zadovoljstva in stališč z/do IS.
Neto koristi	Obseg, v katerem IS prispeva k uspešnosti posameznikov in podjetij.	Povečanje produktivnosti, zmožnosti odločanja, povečanje prodaje, znižanje stroškov.

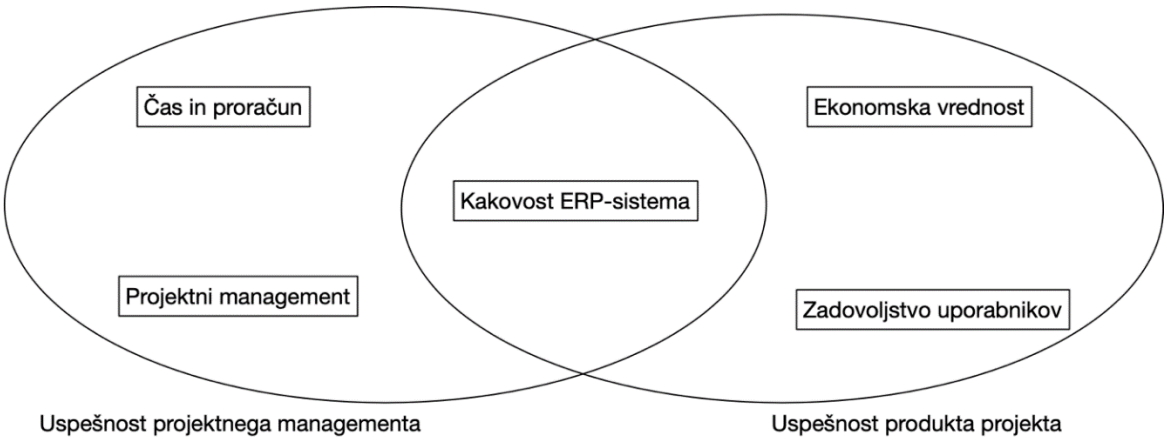
Prirejeno po Petter, DeLone & McLean (2013).

Za celostno ocenjevanje uspešnosti projekta pa je potrebno poleg evalvacije informacijske rešitve dodati še druge dimenzije, med njimi tudi projektni management. Gollner in Baumane-Vitolina (2016) v svojem modelu meritve uspešnosti projektov ERP ločujeta med uspešnostjo projektnega managementa in uspešnostjo produkta projekta. Le obe perspektivi skupaj lahko podata celostno sliko uspešnosti.

Uspešen informacijski sistem označuje npr. funkcionalnosti, ki uporabniku podajo uporabne in natančne informacije, in to enostavno/fleksibilno; to prinaša uporabniku korist v smislu učinkovitosti pri delu in ima za podjetje ekonomsko vrednost. Vidne so pozitivne spremembe v podjetju, ki jih vidimo tudi v zadovoljstvu uporabnikov in vseh deležnikov projekta. Pri meritvi uspešnosti projektnega managementa pa se osredotoča bolj na sam proces in doseganje pri projektu zastavljenih ciljev; konkretno to lahko pomeni doseganje časovnih in stroškovnih okvirov, kot tudi kvalitetno izpeljan proces projektnega managementa ter doseganje specifikacij (obseg) in zadovoljitev potreb deležnikov (Gollner & Baumane-Vitolina, 2016).

Gollner in Baumane-Vitolina (2016) sta zaključila, da je uspešnost projekta možno meriti z naslednjimi petimi dimenzijami: **projektni management**, **čas in proračun**, **kvaliteta sistema ERP**, **ekonomska vrednost** in **zadovoljstvo uporabnikov**. Kot je razvidno na sliki 5, definirata dve skupini ali vidika kriterijev: uspešnost projektnega managementa in uspešnost produkta projekta.

Slika 5: Dimenzije uspešnosti informacijskega sistema



Prirejeno po Gollner & Baumane-Vitolina (2016).

Dimenzija projektnega managementa ima med petimi dimenzijami največjo težo. Vključuje kvaliteto procesa projektnega managementa, skladnost s specifikacijami, zadovoljstvo deležnikov projekta in kvaliteto storitve/podpore. Dober projektni management se kaže v dobro opredeljenem obsegu, prostem pretoku informacij in transparenci delovanja, učinkovitih in takojšnih poteh eskalacije, kadar je to potrebno, transparentnem managementu tveganj in nebirokratski podpori članom tima (Gollner & Baumane-Vitolina, 2016).

1.5.2 Faktorji uspešnosti projekta

Z identifikacijo (in upoštevanjem) dejavnikov ali faktorjev, ki pozitivno vplivajo na uspešen izid projekta, se lahko izboljša možnosti ugodnega rezultata. Faktorjev je veliko in glede na kontekst, tip projekta, podjetje ali zadane cilje različno močno vplivajo na verjetnost uspešnosti projekta.

Petter, DeLone in McLean (2013) so pri svoji analizi 140 študij našli 43 spremenljivk, ki lahko vplivajo na uspešnost informacijskega sistema, in jih razdelili v pet kategorij faktorjev (osnovano na Leavitt diamantu organizacijskih sprememb): **lastnosti nalog, lastnosti uporabnikov, družbene lastnosti, lastnosti projekta in lastnosti organizacije**. Kategorije faktorjev, njihove spremenljivke in obseg vpliva so prikazani v tabeli 3. Medtem ko v kategoriji družbenih lastnosti ni bilo mogoče potrditi nobenega faktorja, ki vpliva na uspešnost informacijskega sistema, so znotraj preostalih kategorij potrdili 15 faktorjev ali neodvisnih spremenljivk, ki vplivajo na celostno uspešnost projekta.

Tabela 3: Faktorji vpliva na uspešnost projekta razvoja informacijskega sistema

Kategorija	Spremenljivka	Vpliv na uspešnost informacijskega sistema
Lastnosti naloge	Kompatibilnost naloge (konsistentnost tehnologije in delovnih procesov).	Srednje močen vpliv.
	Težavnost naloge.	Srednje močen negativen vpliv (manjša težavnost naloge pomeni večjo verjetnost uspeha).
Lastnosti uporabnikov	Odnos do tehnologije.	Srednje močen vpliv.
	Zadovoljstvo pri uporabi.	Močen vpliv.
	Zaupanje.	Močen vpliv.
	Uporabnikova pričakovanja (realistična pričakovanja).	Močen vpliv.
	Vloga v podjetju.	Srednje močen vpliv.
Lastnosti projekta	Vpletenost uporabnikov.	Srednje močen vpliv.
	Odnos do razvijalcev.	Srednje močen vpliv.
	Strokovno znanje področja.	Srednje močen vpliv.
Lastnosti podjetja	Podpora vodstva.	Srednje močen vpliv.
	Procesi upravljanja (npr. procesi organizacijske kulture, administracije, upravljanja sprememb).	Srednje močen vpliv.
	Zunanja motivacija (npr. pritiski ali finančne spodbude iz strani podjetja).	Močen vpliv.
	Organizacijska kompetenca (IT-znanje vodstva).	Srednje močen vpliv.
	IT-infrastruktura.	Srednje močen vpliv.

Prirejeno po Petter, DeLone & McLean (2013).

Kar nekaj faktorjev sovpada z vrednotami in praksami, ki jih vključuje metoda Scrum. Npr. spremenljivki vpletenost uporabnikov ter dober odnos med uporabniki/naročnikom in razvijalci sta pozitivno povezani z uspešnostjo informacijskega sistema in sta hkrati pomemben aspekt metode Scrum. Nadaljnji faktorji, ki prispevajo k uspešnemu zaključku IT-projekta in jih lahko najdemo tudi pri metodi Scrum, spadajo pod okrilje učinkovite komunikacije. Scheucher (2017) prepozna učinkovito komunikacijsko strategijo kot ključno

za uspešno implementacijo projekta. Poudarja pomembnost dnevnih sestankov s timom, vključenost in komunikacijo z deležniki ter vpletenost in podporo najvišjega vodstva. Scheucher (2017) v svoji kvalitativni analizi strategij, ki prispevajo k uspešnosti projekta, identificira naslednje faktorje, porazdeljene v štiri širše teme:

- Učinkovita komunikacija pozitivno vpliva na verjetnost uspeha projekta. Vključuje faktorja vpletenost deležnikov in najvišjega vodstva. Veliko vlogo pa igra tudi obstoj dnevnih sestankov.
- Projektno planiranje pred začetkom implementacije je druga tema. Vključuje analizo tveganj, ki kasneje omogoča učinkovito izogibanje in zmanjševanje tveganj; izbor pravih ljudi za delo na projektu, pomembne so tako kompetence in izkušnje kot zmožnost ljudi za učinkovito skupno delo ter vnaprejšnji izbor metrik za ustrezno meritev napredka projekta.
- Pod temo ustreznega managementa izvedbe projekta spada faktor vodstvene sposobnosti projektnega managerja. Projektni manager, ki je dostopen, spodbuja komunikacijo, ustvarja timsko vzdušje in jasno komunicira obseg, proračun in časovnico (angl. timing) projekta izboljša možnosti uspeha. Prav tako je pomembno upravljanje sprememb, saj te lahko pomenijo neuspeh projekta. Analizi tveganj v prejšnji točki sledi aktivno zmanjševanje tveganja tekom implementacije. Prav tako pa mora projektni manager na podlagi definiranih metrik proaktivno spremljati proces in napredek projekta, če želi težave zaznati zgodaj.
- Tudi zaključek projekta ima vpliv na uspešnost. Tukaj avtor navaja faktorja zadovoljstvo naročnika in shranjevanje znanja, pridobljenega pri projektu.

Pri marsikaterem faktorju, ki po Scheuchnerju (2017) vpliva na večjo verjetnost uspešnosti projekta, lahko vidimo paralele z agilnimi metodami oz. metodo Scrum:

- vpletenost deležnikov v projekt,
- pogosta in učinkovita komunikacija z naročnikom,
- poudarek na redni dnevni komunikaciji znotraj tima,
- podpora vodstva je pomemben faktor, to predpostavlja zaupanje in motiviranje,
- uspešen management sprememb,
- aktivno zmanjševanje tveganja,
- pomembnost zadovoljstva naročnika,
- upravljanje znanja ob koncu projekta.

2 AGILNE METODE

2.1 Razvoj agilnih metod

Tradicionalne metode managementa informacijskih projektov niso več dovolj učinkovite za današnje projekte. Freedman (2009) omenja, da veliko projektov preseže stroškovne in

časovne omejitve projekta ter na koncu ne prinese predvidenih rezultatov. Možna razlaga za takšno stanje so zmotne predpostavke. Avtor navaja naslednje:

- predpostavka, da je planiranje pri tako velikih projektih sploh mogoče,
- predpostavka, da se je pred spremembami pozno v procesu mogoče zaščititi,
- predpostavka, da je sploh smiselno, da se velike projekte natančno definira in splanira zgodaj v projektu.

Eden izmed temeljev agilnih metod je Humphreyev princip negotovosti zahtev, ki pravi da "za nov programski sistem zahteve ne bodo popolnoma znane, dokler ga ne bodo uporabniki dejansko uporabili" (Freedman, 2009). Iz tega stavka so razvidni zametki agilnih metod, ki temeljijo na sprejemanju sprememb zahtev tudi pozno v projektu in na pogosti oddaji programske opreme v pregled in uporabo naročniku. Glede tradicionalne slapovne metode managementa projektov pa Freedman (2009) pravi, da "če uporabniki ne morejo predvideti, kaj dejansko želijo, dokler tega ne vidijo, napovedovanje in planiranje IT-projektov ni mogoče in je zaščita projektov pred spremembami v procesu razvoja nepraktična, potem so ideje za obstoječimi slapovnimi metodami očitno pomanjkljive."

Tradicionalnim metodam in procesom razvoja informacijskih sistemov je bila očitana pretirana birokracija, nefleksibilnost, počasnost/predolgotrajen razvoj, razvoj neuporabnih izdelkov in izguba cilja. Iz teh očitkov se je razvilo razmišljanje o »lažjih metodah«. Te naj bi omogočale hitrejše reagiranje na spremembe v okolju in s tem povezane spremembe zahtev. Tako se je 17 avtorjev in zagovornikov teh metod februarja 2001 dobilo na dvodnevnem sestanku v smučarskem letovišču Snowbird in sestavilo t. i. manifest za agilni razvoj programske opreme. V njem so opredelili 4 skupne vrednote in 12 principov agilnih metod (Beck in drugi, 2001).

2.2 Temeljne vrednote in principi agilnih metod

2.2.1 Manifest agilnega razvoja programske opreme

Beck in drugi (2001) sestavijo sledeči manifest agilnega razvoja programske opreme:

"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 z naročnikom pred pogodbenimi pogajanji
- Odziv na spremembe pred togim sledenjem načrtom

Z drugimi besedami, če tudi cenimo dejavnike na desni, vseeno bolj cenimo tiste na levi."

Pod manifestom je podpisanih 17 avtorjev, ki so se v svojem delu ukvarjali ali celo razvili kakšno agilno metodo razvoja programske opreme: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland in Dave Thomas. Manifest vključuje 4 osnovne vrednote in 12 principov agilnih metod opisanih spodaj.

2.2.2 Temeljne vrednote agilnega razvoja programske opreme

4 vrednote agilnih metod so z namenom oblikovane abstraktno, saj jih moramo razumeti kot smernice in ne stroga pravila. Poudarjajo pomen kontinuirane komunikacije, tako med člani tima kot z deležniki, in pa pomen razvoja delujoče in uporabne programske opreme kot glavnega kriterija uspeha. Oblika vrednot namenoma vključuje dva vidika (orodja pred procesi, sodelovanje z naročnikom pred pogodbenimi pogajanji itn.), saj se poudarja, da agilne metode ne zavračajo vidika na desni, ampak zgolj dajejo toliko večji pomen vidiku na levi (Koch, 2005).

2.2.2.1 Posamezniki in interakcije pred procesi in orodji

Motivirani posamezniki so pri agilnih metodah ključni. Motivacija pa izvira iz optimalnega delovnega okolja in zaupanja. Koch (2005) predpostavlja, da motivirani posamezniki naredijo več kot samo sledijo navodilom nadrejenega, ampak znajo uporabiti svojo lastno strokovno oceno, da izberejo najbolj optimalno pot do cilja. Motivacija posameznikov pomeni tudi večjo vpletenost, odgovornost in zavezanost delu ter posledično večje zadovoljstvo članov tima. Po agilnih metodah se v podjetju ustvari okolje, ki posameznikom omogoča čim boljši izkoristek svojih sposobnosti in kapacitet.

Agilne metode temeljijo na upoštevanju ljudi kot ključnega faktorja uspeha projektov razvoja programske opreme, vendar to s pomočjo orodij in procesov. Slednji niso zanemarjeni ali izključeni zgolj postavljeni v podporno vlogo. Kot pravi Koch (2005), imajo ljudje pomanjkljivosti in tudi najboljši tim sam ni dovolj za uspešno izvedbo projekta, potrebujejo orodja in procese, ki so jim v podporo.

Cobb (2015) to vrednoto vidi kot povratni odgovor na pristope projektnega managementa, ki temeljijo na »kontroli« in rigidnih procesih. Ti pristopi so pogosto neosebni in ne upoštevajo potreb posameznikov. Agilni pristop vključuje »mehki« management, kjer so timi samoorganizirani, torej imajo posamezniki vpliv na organizacijo svojega dela. Procesi so fleksibilni in prilagodljivi. Organizacijska struktura je manj hierarhična in bolj ravna. Koch (2005) poudarja, da je v agilni organizacijski strukturi tim partner tako vodstvu kot tudi naročniku. Tim je zmožen sam sprejemati odločitve, kar pa vodi ne samo k bolj motiviranim posameznikom, ampak tudi k poslovni odličnosti.

Pomemben del managementa projekta je komunikacija, ki pa s povečevanjem kanalov komuniciranja lahko postane neučinkovita. Po agilni metodi je najbolj učinkovita komunikacija neposredna komunikacija, torej komunikacija v živo. Dokumentacija je sicer v določenih primerih pomembna, ampak Koch (2005) poudarja, da ta ne sme neupravičeno zamenjati zelo učinkovitega pogovora. Iz vseh teh razlogov se spodbuja, da je tim fizično na isti lokaciji.

Zadnji princip manifesta, ki se nanaša na ljudi in interakcije, je spodbujanje stalnega ritma dela. Posamezniki so bolj učinkoviti, kadar so nadure omejene. Sprinti oz. časovno omejeni cikli in jasen pregled napredka omogočajo konsistenten ritem dela (Koch, 2005).

2.2.2.2 Delujoča programska oprema pred vseobsežno dokumentacijo

Dokumentacija v projektu igra pomembno vlogo podpore neposredni komunikaciji in zapisa pomembnih informacij za kasneje. Tudi pri agilnih metodah se v dokumentaciji daje vrednost, vendar se hkrati poudarja, da le-ta ne sme biti preobsežna ali sama sebi namen. Koch (2005) opozarja na sledeče lastnosti dokumentacij, ki zmanjšujejo njeno učinkovitost in uporabnost:

- dokumentacija nima jasnega namena,
- dokumentacija nima jasno definiranega ciljnega občinstva,
- dokumentacija je preveč ali premalo natančna glede na namen in občinstvo,
- dokumentacijo se vzdržuje kljub temu, da nima več uporabne vrednosti.

Cobb (2015) poudarja, da pretiran pomen dokumentacije ovira komunikacijo; npr. pri projektih se z zanašanjem na dokumentacijo odpira vrata nesporazumom. Specifikacija zahtev na začetku je lahko zgolj omejeno natančna in testiranje zgolj po teh napisanih kriterijih posledično zavajajoče.

Bolj pomembno kot dokumentacija, ki mora igrati podporno vlogo, pa je dejanski produkt projekta, torej korekten razvoj delujoče programske opreme. Pri agilnih metodah je v ospredje postavljeno zadovoljstvo naročnika, ki se ga dosega z zgodnjo in konstantno oddajo programske opreme, ki deluje in ima vrednost (Koch, 2005). Avtor nadaljuje, da pogosta predaja programske opreme zmanjšuje možnosti nezadovoljstva naročnika, saj se morebitna neskladja pričakovanj odkrijejo prej, kot kadar se programsko opremo odda naročniku na koncu projekta; prav tako pa je možno napredek meriti s samim produktom oz. delujočo programsko opremo.

2.2.2.3 Sodelovanje z naročnikom pred pogodbenimi pogajanji

Agilne metode temeljijo na tesnem sodelovanju z naročnikom in uporabniki, ki so preko bolj natančnega vpogleda v projekt in pogoste oddaje programske opreme bolj vpleteni in informirani. Kot pravi Cobb (2015), je v negotovih okoljih kolaborativni pristop bolj

učinkovit, vendar zahteva določeno stopnjo zaupanja med razvijalci in naročnikom. Koch (2005) razlaga, da se tekom projekta ustvarja znanje, ki zamenja ali dopolni prvotne predpostavke. S tem usmeri projekt v smer, ki vodi k zadovoljstvu naročnika.

Vendar je to ustvarjanje novih znanj o funkcionalnostih lahko izvor nezaželene širitve obsega projekta (angl. scope creep). Pogodbe so poskus kontrole teh stroškov in obsega. Koch (2005) navaja dve strategiji agilnih metod za zajezitev problema širitve obsega, s katerimi lahko naprej zagovarja spodbujanje čim večje stopnje sodelovanja naročnika pri projektu:

- Agilne metode predpostavljajo tesno sodelovanje z naročnikom. To pomeni, da se zaradi večjega razumevanja tako spreminjajočega se sistema kot potreb naročnika iz obeh strani, funkcionalnosti ne samo dodajajo, ampak tudi odvzemajo.
- Ker naročnik sodeluje pri planiranju in postavljanju prioritet, morebitno odgovornost za povečanje obsega projekta nosi tudi sam. Naročnik odloča o tem, katere funkcionalnosti se bodo dodale in kakšni bodo potrebni kompromisi – povečanje proračuna, podaljšanje projekta in podobno.

Pogodbe so kljub tesnemu sodelovanju pomembne, ampak pogodbena pogajanja po agilnih metodah naj ne bi nadomestila sprotnega sodelovanja naročnika v projektu. Fowler in Highsmith (2001) pravita, da pogodbe definirajo omejitve in pogoje za delo, vendar pa lahko razvojni tim le skozi sodelovanje z naročnikom razume in naredi programsko opremo, ki ustreza željam naročnika.

Prav tako Koch (2005) opozarja, da pogodba ne sme omejevati sodelovanja z naročnikom. Med projektom se ustvarja znanje, ki vodi do prilagoditev v projektu; preveč restriktivna pogodba lahko zaradi potrebnih sprememb vodi v dodatno delo, administracijo, stroške, npr. za odvetniške storitve. S pogodbo naj bi se pogodbeni strani tudi dogovorili za stopnjo, do katere je mogoče prilagajati obseg, stroške in funkcionalnosti brez spreminjanja pogodbe.

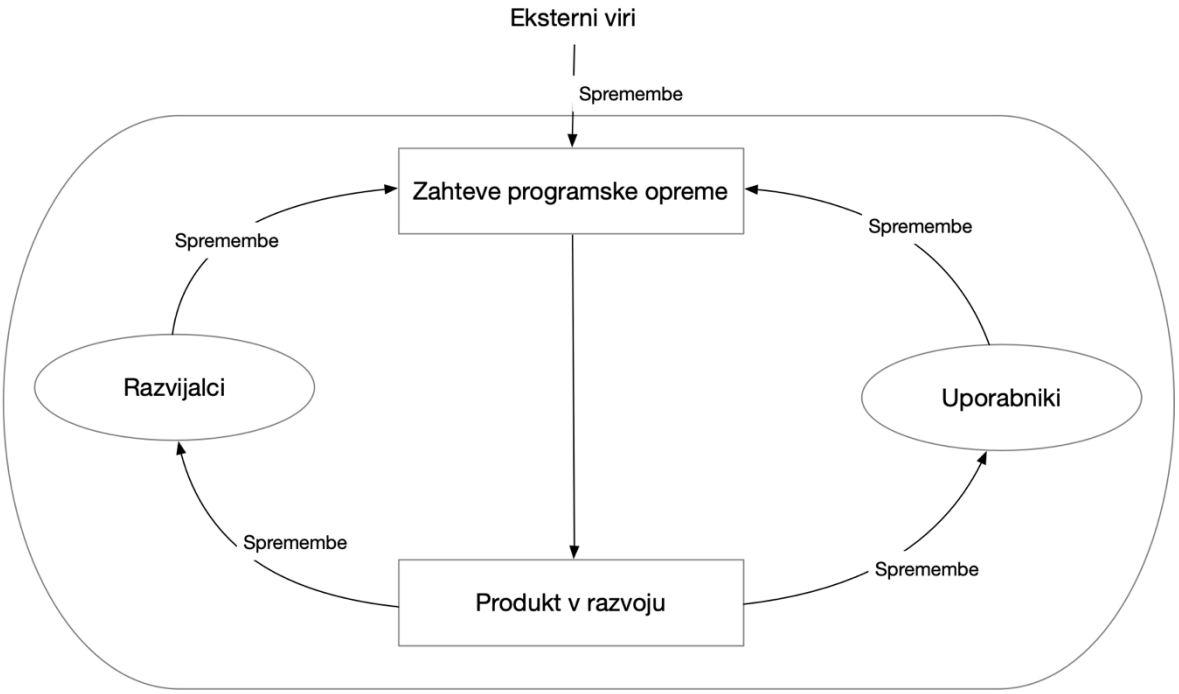
2.2.2.4 Odziv na spremembe pred togim sledenjem planu

Vse večja nepredvidljivost je ena največjih izzivov nove ekonomije, menita Fowler in Highsmith (2001), turbulence današnjega časa povzročajo spremembe, ki jih je možno dojemati ali kot grožnjo ali kot priložnost. Pri agilnih metodah se predpostavlja, da so spremembe neizogibne in se jih zato sprejema, to je v nasprotju s tradicionalnimi pristopi, ki zagovarjajo natančno planiranje in sledenje planu. Koch (2005) razlaga, da je to možno, ker agilne metode vključujejo delovanje v krajših ciklih (1−4 tednov) in pogosta manjša planiranja. Tako se lahko hitreje reagira na spremembe.

Spremembe delimo na eksterne in interne. Eksterne spremembe se nanašajo na spremembe v okolju, npr. sprememba zakonodaje, medtem ko se interne spremembe nanašajo na

spremembe, ki so posledica učenja tekom projekta. Slika 6 prikazuje vpliv obeh tipov sprememb na zahteve programske opreme (Koch, 2005).

Slika 6: Vpliv sprememb na zahteve



Prirejeno po Koch (2005).

2.2.3 Principi agilnih metod

Poleg štirih vrednot avtorji (Beck in drugi, 2001) v manifestu agilnih metod definirajo tudi 12 principov, ki še natančneje orišejo bistvo agilnih metod:

- Naša najvišja prioriteta je zadovoljiti naročnika s čimprejšnjo izdelavo kvalitetne programske opreme.
- Sprejemati spremembe zahtev tudi v poznih fazah razvoja. Agilni procesi tovrstne spremembe izkoristijo za izboljšanje konkurenčnosti naročnika.
- Delujočo programsko opremo izdajati pogosto, znotraj obdobja nekaj tednov, do nekaj mesecev, s preferenco po krajšem časovnem okvirju.
- Poslovneži in razvijalci morajo skozi celoten projekt dnevno sodelovati.
- Projekte gradimo okrog motiviranih posameznikov. Omogočimo jim delovno okolje, nudimo podporo in jim zaupamo, da bodo svoje delo opravili.
- Najboljša in najučinkovitejša metoda posredovanja informacij razvojnemu timu in znotraj tima samega je neposreden pogovor.
- Delujoča programska oprema je primarno merilo napredka.
- Agilni procesi promovirajo trajnostni razvoj. Sponzorji, razvijalci in uporabniki morajo biti zmožni stalnega ritma za nedoločen čas.

- Nenehna težnja k tehnični odličnosti in učinkovitemu načrtovanju izboljša agilnost.
- Preprostost, umetnost zmanjševanja količine nepotrebnega dela je bistvena.
- Najboljše arhitekture, zahteve in načrti izhajajo iz tistih timov, ki so samoorganizirani.
- V rednih časovnih obdobjih tim išče načine, kako postati še učinkovitejši ob rednem prilagajanju svojega delovanja.

Agilne metode težijo k povrnitvi ravnotežja med metodami, pravili in fleksibilnostjo. Kljub vsemu tudi agilne metode vsebujejo pravila, ne zavračajo pomena planiranja in dokumentacije, vendar omejujejo njihov pomen proti drugim vrednotam. Vse z namenom ohranjanja fleksibilnosti oz. zmožnosti hitre in učinkovite odzivnosti na spremembe.

2.3 Pregled agilnih metod

Pod okrilje agilnih metod poleg metode Scrum spadajo tudi naslednje metode: ekstremno programiranje (angl. Extreme Programming, v nadaljevanju XP), funkcijsko voden razvoj (angl. Feature Driven Development, v nadaljevanju FDD), Kanban, Crystal, metoda dinamičnega razvoja sistemov (angl. Dynamic Systems Development Method), metoda adaptivnega razvoja programske opreme (angl. Adaptive Software Development) in mnoge druge. Nekaj najbolj razširjenih podrobneje opisujem spodaj.

2.3.1 Ekstremno programiranje

Ustanovitelj metode je Kent Beck, označuje jo kot lahko metodo za majhne do srednje velike time, ki programsko opremo razvijajo pri zgolj ohlapno definiranih in vedno spreminjajočih se zahtevah. Poudarja se, da principi niso nekaj revolucionarnega, ampak zgolj preizkušeni in zaupanja vredni principi razvoja, ki so poneseni na višjo, ekstremno stopnjo. Pet vrednot, ki se jih po metodi še posebej poudarja, so komunikacija (na vseh nivojih in med vsemi deležniki), povratne informacije, preprostost (pred ustvarjanjem kompleksnosti z nepotrebnimi funkcijami), pogum (predvsem za sprejemanje sprememb in posledične spremembe kode) in spoštovanje (ki omogoča zaupanje in dajanje/sprejemanje odgovornosti) (Fitzgerald, Hartnett & Conboy, 2006; Holcombe, 2008).

Koch (2005) in Holcombe (2008) navajata naslednje temeljne prakse metode:

- Igra načrtovanja je kolaborativni in ponavljajoči se proces, s katerim se prične razvoj vsakega inkrementa. Management, tehnični tim in uporabniki poslovne potrebe prevedejo v dosegljiv načrt. To vključuje med drugim pripravo, ocenitev in izbor zgodb, občasni pregled poslovnih ciljev z uporabniki (in naročnikom), razvoj testov ter ocenitev stroškov in drugih virov.
- XP vključuje majhne in pogoste izdaje (angl. release). Torej razvoj najmanjšega možnega dela sistema, pri katerem je še možno prikazati vrednost uporabniku.

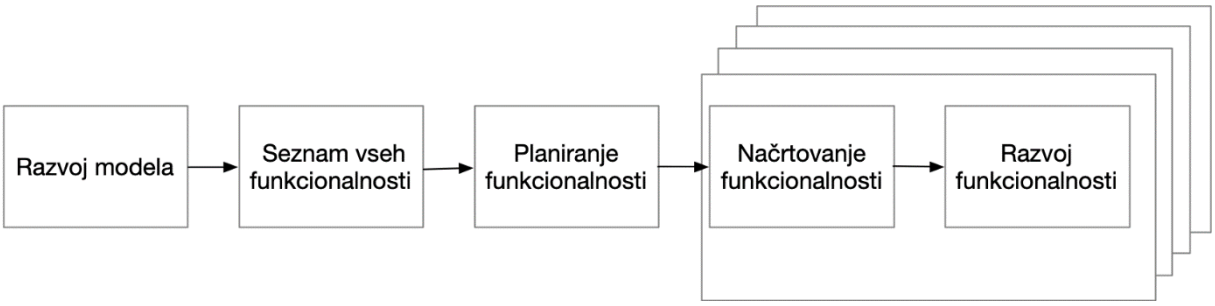
- Pod »metaforo« XP-programerji razumejo splošen koncept sistema, ki ga želijo zgraditi. Je vizija sistema kot ga razumejo uporabniki in razvojni tim. Metafora ostaja relativno nespremenjena tekom projekta in kot vizija vodi razvijalce. To pa ne velja za »zgodbe«, ki prikazujejo posamezne funkcionalnosti in se z novimi informacijami tekom razvoja spreminjajo in prilagajajo.
- Preprostost pri razvoju je filozofija opuščanja nepotrebnih funkcionalnosti. To so tiste, ki niso nujno potrebne za razvoj trenutne zgodbe in bi lahko zgolj po nepotrebnem vnesle v sistem kompleksnost. Funkcionalnosti, ki jih bi mogoče rabili v prihodnosti, se ne razvija »na zalogo«. Popravkom se ne da izogniti, je pa lažje popravljati preprost kot kompleksen načrt.
- Celoten proces razvoja je centriran okoli testiranja; le tako je mogoče kontinuirano spremljanje napredka. Pred začetkom programiranja funkcionalnosti se razvije skupek testov. Pri tem s svojim znanjem procesov podjetja pomagajo tudi uporabniki.
- Preurejanje in optimizacija programske kode (angl. refactoring) je prilagajanje kode brez spremembe funkcionalnosti z namenom ustvarjanja kode, ki je preprostejša, bolj razumljiva in lažja za vzdrževanje.
- Programiranje v paru je ena izmed najbolj znanih značilnosti XP-programiranja. Pri razvoju se formirajo pari programerjev. To omogoča, da je med programiranjem prisoten tudi pogovor in pregled ustreznosti. En v paru programira, drugi pregleduje in razmišlja o nadaljnjih korakih. Kontinuirani pregled kode pomeni manj napak in omogoča, da so razlogi za določen način implementacije ves čas odprti za razpravo in morebitne izboljšave.
- Koda je v kolektivnem lastništvu, kar pomeni, da si kode noben posameznik ne lasti. Spreminja jo lahko vsak programer, seveda skladno z dogovorjenimi standardi kodiranja. Cilj je, da vsak programer razume vsak del kode, njen namen in vlogo.
- Kontinuirana integracija pomeni, da se nova koda integrira takoj, posamično, in ne skupaj na koncu; tako je sistem takoj posodobljen z novo funkcionalnostjo. Predpogoj za integracijo so uspešno prestani testi, po integraciji pa je koda sprejeta le, če so tudi funkcionalni testi sistema uspešni.
- Po XP metodi je definiran največ 40-urni delavnik. Nadure so nezaželene, saj mora biti omogočeno, da programerji trajnostno zmorejo predviden tempo razvoja.
- Če je naročnik fizično prisoten na lokaciji razvoja, to izboljša povratno zanko informacij. Tako je možno bistveno hitreje razrešiti nejasnosti ter dobiti odgovore na vprašanja.
- Standardi kodiranja se definirajo na začetku projekta in vključujejo pravila in dogovore glede skupnega dela, npr. definicija stila kodiranja, pravila poimenovanja razredov in metod, način testiranja in oblika uporabniških zgodb.

2.3.2 Funkcijsko voden razvoj

V nasprotju z drugimi agilnimi metodami, se daje pri metodi FDD v zgodnjih fazah razvoja informacijske rešitve večji poudarek na načrtovanje in planiranje. Na začetku procesa se

definira model sistema in pripravi lista funkcionalnosti, ki se nato razvijajo med posameznimi iteracijami. Slika 7 prikazuje proces po tej metodi.

Slika 7: FDD-proces



Prirejeno po Koch (2005).

Koch (2005) metodo opiše preko osmih praks:

- Objektno modeliranje pomeni, da se na začetku procesa pripravi model razvijajočega se sistema, katerega namen ni natančen prikaz sistema, ampak razkriva želje in predpostavke deležnikov in predstavlja načrt nadaljnjega razvoja. Model se tekom razvoja posodablja, tako da vedno predstavlja aktualno predstavo končne rešitve.
- Razvoj poteka po funkcionalnostih. V vsaki iteraciji se razvija ena ali več funkcionalnosti. Funkcija je definirana z vrednostjo za naročnika in možnostjo razvoja v do dveh tednih.
- V nasprotju z skupnim lastništvom pri metodi XP, se pri metodi FDD predvideva, da ima vsak razred svojega »lastnika«, ta natančno pozna razred in je tako bolj učinkovit pri implementaciji sprememb v tem razredu. Vendar pa avtor opozarja, da skupaj s tem pride tudi tveganje negativnih posledic v primeru izpada lastnika razreda.
- Timi so organizirani po funkcionalnostih. Za razvoj vsake funkcionalnosti se sestavi tim, ki vključuje lastnika funkcionalnosti (glavni programer) in lastnike posameznih razredov, na katere razvoj funkcije vpliva. Timi so dinamični del metode FDD, saj se lahko oblikujejo in razpuščajo tudi dnevno glede na potrebe. Hkrati deluje več timov in vsak programer je lahko aktiven v več timih. Implementacija funkcionalnosti preko lastnikov razredov naj bi bila tako hitrejša in bolj učinkovita.
- Inšpekcije pomenijo natančen pregled načrta in kode in so pomembno orodje za zmanjševanje tveganja, ki ga prinaša individualno lastništvo razredov. Z njimi se vzdržuje visoke standarde kodiranja.
- Predvidene so kontinuirane graditve (angl. build) funkcionalnosti v sistem, ki je s tem vedno posodobljen z že razvitimi funkcionalnostmi. Frekvenca integracij naj bi bila najmanj tedenska, če je le možno, dnevna.
- V praksi je tudi management konfiguracij. Metoda sicer ne definira specifik upravljanja konfiguracij, ampak zaradi dinamike timov in visoke frekvence implementacij prepozna potrebo po njej. Priporoča se tako strog management konfiguracij, kot je primerno za velikost, kompleksnost in obseg projekta.

– Za poročanje in prikaz statusa projekta se uporablja enačbo 1, ki pri izračunu upošteva zahtevnost razvoja posameznih funkcionalnosti.

$$\text{Status projekta} = \frac{\sum \text{vrednosti vseh funkcionalnosti}}{\text{število vseh funkcionalnosti na seznamu}} \quad (1)$$

Vsaka funkcionalnost vsebuje mejnike, ki pa so uteženi glede na čas potreben za doseganje mejnika. Kot prikazuje enačba 2, se vrednost funkcionalnosti izračuna s seštevkom vseh uteži že končanih mejnikov. Status projekta je seštevek uteži doseženih mejnikov pri vseh funkcionalnostih deljeno s številom funkcionalnosti.

$$\text{Vrednost funkcionalnosti} = \sum \text{uteži končanih mejnikov funkcionalnosti} \quad (2)$$

Vrednost funkcionalnosti, na kateri se še ne dela = 0,00
Vrednost funkcionalnosti, ki je v delu = med 0,00 in 1,00
Vrednost funkcionalnosti, ki je zaključena = 1,00

2.3.3 Kanban

Kanban ima korenine v vitki proizvodnji (angl. lean manufacturing) in temelji na konstantni optimizaciji procesov in zmanjševanju izgub. Gradi na principu pravega trenutka (angl. just in time), ki definira tok dela tako, da so predhodne aktivnosti procesa sprožene šele, ko je možna obdelava v kasnejših aktivnostih. Cilj je tekoč potek produkcije z minimalnim številom izgub. V kontekstu kanbana izgube predstavljajo zgolj delno opravljeno delo, razvoj nepotrebnih funkcij, dodatni nepotrebni procesi in pogosto menjavanje med nalogami. Prednosti uporabe Kanban metode je v zmanjševanju potrebnega časa do izdaje (angl. lead time), konsistentnem toku izdaj, večji kvaliteti programske opreme, boljšemu razumevanju procesa, izboljšani komunikaciji, višjem zadovoljstvu naročnikov in večji motivaciji ekipe razvijalcev (Ahmad, Markkula & Oivo, 2013).

Measley (2015) Kanban definira kot alternativno pot k agilnosti in ne kot metodo ali proces. Vloge in procesi po Kanbanu niso opredeljeni, ampak se gradijo na konceptu evolucijskih sprememb, ki ne glede na vse ostalo stremijo h konstantni optimizaciji toka dela in rezultatov.

Measley (2015) ter Ahmad, Markkula in Oivo (2013) navajajo šest osnovnih praks po Kanbanu:

– Na veliki tabli, fizične ali elektronske oblike, uporabniki Kanbana vizualizirajo svoje delo. Lističi z zahtevami so umeščeni v eno izmed definiranih faz procesa, od seznama zahtev (angl. backlog) do zaključenih zahtev. To omogoča poleg samega pregleda dela, jasno razvidnih prioritet in morebitnih ozkih grl, tudi lažjo identifikacijo priložnosti za optimizacijo procesa.
– Po Kanbanu so zahteve v delu (angl. work in progress) omejene. Definira se število zahtev, ki jih razvojni tim lahko opravlja hkrati; šele, ko je ena izmed zahtev zaključena,

lahko razvojni tim začne z novo zahtevo. Omejitev zahtev znotraj razvoja zmanjšuje stroške koordinacije, povečuje osredotočenost na posamezne zahteve v nasprotju z manj učinkovitim delom na več zahtevah hkrati (angl. multi-tasking), zmanjšuje čas do izdaje (angl. lead time) in povečuje kvaliteto produkta. Prav tako so težave v procesu hitreje odkrite in odpravljene ter tim dela s tempom, ki ga lahko trajnostno ohranja. Število zahtev, na katerih se dela istočasno, pa lahko variira, saj se po Kanbanu spodbuja eksperimentiranje s številom in iskanje optimalnega obsega.

- Pravila in procesi so dokumentirani. Cilj je v jasnosti in učinkovitosti dela. Z vnaprej definiranimi in dokumentiranimi pravili, npr. odločanja v neki situaciji, lahko tim brez dodatnega usklajevanja pravilo takoj uporabi.
- Prehodi med koraki procesa so merjeni in ustvarjajo sliko toka dela skozi čas s ciljem identifikacije priložnosti optimizacije procesa.
- Zanke povratnih informacij so implementirane na vseh nivojih. To omogoča učenje iz učinkov morebitnih sprememb na procese in s tem identifikacijo potencialnih optimizacij.
- Goji se kultura stalne optimizacije kot odgovornosti vseh vpletenih.

3 METODA SCRUM

Vse več podjetij potrebuje hitrost in fleksibilnost za razvoj novih produktov, ki predstavljajo čedalje večji delež prihodkov in profita. To se nanaša tako na IT-produkte kot produkte drugih disciplin. Pod »starim« pristopom se je razvoj premikal sekvenčno iz ene faze v drugo. Vloge so bile segmentirane in različni oddelki so si pri prehodu v naslednjo fazo predali projekt. Projekt je potoval linearno od enega oddelka do drugega. Nasprotno Takeuchi in Nonaka (1986) zagovarjata, da je za hitrejši in učinkovitejši razvoj potreben pristop, ki omogoča kolaboracijo članov tima od začetka do konca projekta, kjer se faze razvoja prekrivajo in je možno iterativno eksperimentiranje in učenje.

Metoda Scrum je ena izmed najbolj razširjenih agilnih metod v IT-industriji. Temelji na samoorganiziranih večfunkcionalnih timih in se zaradi pozitivnih učinkov praks na dinamiko in učinkovitost timov ter poudarkom na kolaboraciji hitro razširja tudi na druge discipline. Izraz Scrum uporabita Takeuchi in Nonaka že leta 1986 v članku ”The new new development game*“*. Izposodila sta si ga iz ameriškega nogometa, kjer si igralci podajajo žogo in se kot enota premikajo preko igrišča. Poudarja kolaborativno naravo dela. Novi (Scrum) pristop definirata kot celostni pristop z naslednjimi lastnostmi: nestabilnost je samoumevna, samo-organiziranost tima, prekrivajoče faze razvoja, učenje prisotno na več nivojih, zmanjšana kontrola in organizacijski prenos učenja.

Scrum je metoda, ki jo med drugim lahko vidimo kot odgovor na vse večjo preobremenjenosti delovne sile z delom na več projektih simultano in posledično zmanjšano osredotočenostjo na posamične naloge. Rezultat je padec kvalitete izvedbe nalog. Quigley in Pries (2001) tukaj omenjata fanatično večopravilnost (angl. frantic multitasking), ki jo je

obravnaval ameriški psihiater Edward Hallowel. Pri velikem številu simultanih nalog je priotiziranje med njimi oteženo in lahko privede do tega, da oseba »zmrzne« in se ne more več odločiti med dvema možnostima.

Pri metodi Scrum se spodbuja osredotočenost na omejeno število nalog, uporabniških zgodb v omejenem časovnem okviru. Kot pravita Quingley in Pries (2011) je metoda Scrum način, kako priti iz neoptimalnega opravljanja več nalog hkrati v visoko efektivno osredotočanje na eno nalogo in s tem povečati produktivnost.

Timu razvijalcev je naložena veliko večja vloga in odgovornost. V nasprotju z običajno podrobno opredelitvijo nalog se načrtovanje izvedbe posameznih nalog prepušča timu, ki najbolje ve, kaj je potrebno narediti, da se reši dani problem. Tim, ki sledi metodi Scrum, je samoorganiziran ter odločitve sprejema skupaj.

Fitzgerald, Hartnett in Conboy (2006) pravijo, da se Scrum od tradicionalnih metod razlikuje v pogledu, da je za procese, analize, načrte in razvoj značilna nepredvidljivost. Zato je projekt po metodi Scrum odprt za spremembe skozi celoten proces razvoja. Projekt je odprt za spremembe iz okolja, spremembe konkurence, spremembe v časovnih, kvalitativnih ali finančnih okvirih. Kar agilnim metodam omogoča prilagajanje spremembam, je iterativni pristop razvoja, kjer je razvoj razdeljen v mnogo manjših ciklov, na koncu vsakega pa povratna zanka informacij, ki omogoča kontrolo in prilagoditev smeri razvoja.

Visoka vpletenost naročnika med projektom omogoča, da vnaprej natančno določene zahteve niso potrebne. Pogosta komunikacija in pregled opravljenega dela/produkta v proces vključujeta povratno zanko informacij, ki lahko povzroči posodobitev in spremembo prioritet zahtev. Visoka vpletenost naročnika zagotavlja večjo kvaliteto produkta, saj le-ta bolj natančno izpolni namen produkta in zadovolji s tem povezane poslovne potrebe ter kontrolo tekom projekta (Koch, 2005).

Zbrane zahteve so definirane namesto v obsežni dokumentaciji v uporabniških zgodbah (angl. user stories), ki na jasen način razložijo, kaj naj bi določena funkcionalnost uporabniku omogočala. Obsežna dokumentacija je že zaradi relativno hitrih sprememb zahtev med projektom nezaželena (Quigley & Pries, 2011).

Kot vsaka metoda ima tudi Scrum svoje omejitve. Ionel (2008) izpostavi nekaj šibkih točk. Vpletenost uporabnika ali naročnika je lahko pomanjkljivost, če le-ta nima jasne vizije produkta ali ni na voljo za pogoste sestanke in vprašanja. S tem je možno delo v napačno smer in posledično produkt, ki ne zadošča potrebam naročnika.

Prav tako je majhnost tima potencialna šibkost, kadar govorimo o velikih projektih. S ciljem tvorbe majhnih razvojnih timov je projekt razdeljen na več delov. Nastane več manjših projektov vodenih po metodi Scrum in koordinacija med njimi je potencialno zelo zahtevna (Ionel, 2008).

3.1 Vloge po metodi Scrum

Vloge po metodi Scrum so nekoliko drugačne kot pri tradicionalnem managementu projektov. Projektni tim sestavljajo skrbnik metode (angl. Scrum master), produktni vodja (angl. product owner) in razvojni tim. Veliko večjo vlogo in odgovornost po metodi Scrum prevzema razvojni tim sam, ki se ob začetku vsakega cikla zaveže za razvoj določenih uporabniških zgodb. Skrbnik metode zagotavlja, da se proces po metodi Scrum korektno izvaja in da ima tim najboljše možne pogoje za delo. Celostno sliko nad projektom pa ima produktni vodja, ki je tudi zadolžen za upravljanje seznama zahtev produkta.

3.1.1 Skrbnik metode Scrum

Ima pomembno vlogo, saj skrbi za pravilno izvajanje metode Scrum in za nemoten potek dela. Mahnič in Urevc (2012) skrbnika metode opredeljujeta kot varuha, ki varuje razvojni tim pred škodljivimi zunanjimi vplivi, odpravlja morebitne ovire in s tem zagotavlja optimalne pogoje za delo.

Quigley in Pries (2011) navajata naslednje naloge skrbnika metode:

- odpravljanje ovir za tim,
- posredovanje pri nesporazumih,
- zagotavljanje upoštevanja metode Scrum in timskih pravil,
- pridobivanje virov,
- omogočanje timu, da se osredotoča na delo (odpravljanje motenj),
- pregled napredka preko diagrama preostalega dela (angl. burndown chart).

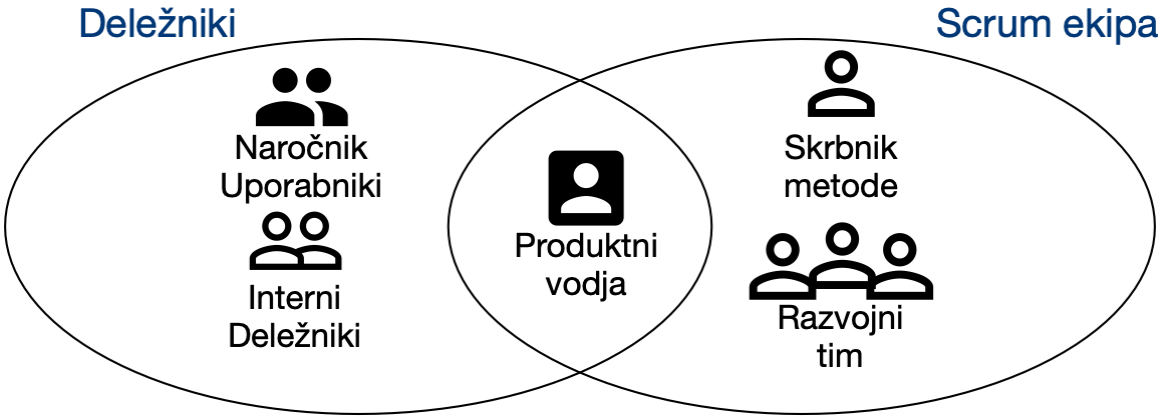
Skrbnik metode timu kot tudi celotnemu podjetju pomaga pri razumevanju in implementaciji praks metode Scrum. Prav tako pomaga podjetju in managementu pri tranziciji/uvedbi metode. Deluje kot moderator pri reševanju težav in spodbuja izboljšave pri uporabi metode Scrum (Rubin & Lichtenberg, 2014).

Naziv projektni manager v metodi Scrum ni več prisoten, prav tako se njegove odgovornosti drugače porazdelijo. Skrbnika metode tako ne moremo enačiti s projektnim managerjem. Skrbnik metode je timu mentor, nad njim nima formalne moči ali avtoritete.

3.1.2 Produktni vodja

Vloga produktnega vodja je osrednji člen pri procesu razvoja produkta in je vezni člen z ostalimi vpletenimi v proces. Kot pravita Rubin in Lichtenberg (2014) mora biti produktni vodja vedno usmerjen v vsaj dve smeri istočasno. Kot je razvidno iz slike 8, na eni strani komunicira z deležniki, med katere štejemo tako vodstvo podjetja kot bodoče uporabnike, in na podlagi njihovih potreb in želja skupaj ustvarjajo vizijo produkta in zahteve.

Slika 8: Produktni vodja kot vezni člen med deležniki in Scrum timom



Prirejeno po Rubin & Lichtenberg (2014).

Na drugi strani sodeluje s projektnim timom in z njimi deli informacije, usklajuje in določa prioriteto zahtev ter skrbi za ustrezne kriterije odobritve razvitih funkcij. Kot ugotavljata Urevc in Mahnič (2012), je njegova vloga ključnega pomena za uspešnost projekta, saj mora imeti produktni vodja jasno vizijo cilja in skladno s tem usmerjati razvoj.

Produktni vodja je skrbnik seznama zahtev produkta (angl. product backlog), ki ga redno dopolnjuje in posodablja glede na razmere na trgu, spreminjajoče se zahteve ali nove informacije. Ima pregled nad celotnim projektom in je razvojnemu timu vedno na voljo za morebitna vprašanja ali pojasnila, vezana na zahteve oz. uporabniške zgodbe.

Rubin in Lichtenberg (2014) naloge produktnega vodje povzameta sledeče:

- sklepanje poslovnih odločitev,
- sodelovanje pri (portfelj, produkt, izdaja in sprint) planiranju aktivnosti,
- upravljanje seznama zahtev,
- definiranje kriterijev sprejemljivosti in skrb za upoštevanje le-teh,
- sodelovanje z razvojnim timom,
- sodelovanje z deležniki.

3.1.3 Razvojni tim

Pri metodi Scrum se še posebej osredotoča na razvojni tim in se ga postavlja v center procesov in praks. Tim ima v metodi Scrum večjo vlogo in več odgovornosti, člani tima so aktivno vpleteni v planiranje in izvedbo sprintov in si delijo odgovornost tudi za učenje in napredek.

Da je možna predvidena visoka stopnja komunikacije med člani tima, tim sestavlja največ 10 članov. Z manjšim timom je možna večja učinkovitost, vsi člani so informirani o stanju in napredku projekta in čutijo višjo zavezanost timu in skupnemu cilju, kot bi to bilo v večjih

timih. Razvojni tim je zadolžen za implementacijo uporabniške zgodbe in je sestavljen tako, da vključuje člane iz vseh področij, potrebnih za uspešen razvoj zahtevane funkcionalnosti (Rubin & Lichtenberg, 2014).

Quigley in Pries (2011) se opirata na Bellmannove atribute za prikaz karakteristik samoorganiziranega tima:

- Ravnovesje. Moje delo je usklajeno z mojim življenjem.
- Inovacija. Moje delo zadovoljuje potrebo po rasti in ustvarjanju.
- Avtentičnost. Večinoma me delo spodbuja, da sem »jaz« in ne igram zgolj neke vloge.
- Doprinos. Skozi moje delo doprinesem majhno, a pozitivno razliko k družbi.
- Pogum. Pripravljen sem zagovarjati moje stališče drugim v podporo idejam in dejanjem.
- Kvaliteta. Moji delovni standardi so visoki in se jih držim.
- Pravičnost. Ljudi sodim po njihovem potencialnem doprinosu k investiciji v našem skupnem delu.
- Zaupanje. Do drugih sem odprt in pozitiven in predvidevam, da se lahko na njih zanašam tako, kot se oni lahko zanesejo name.

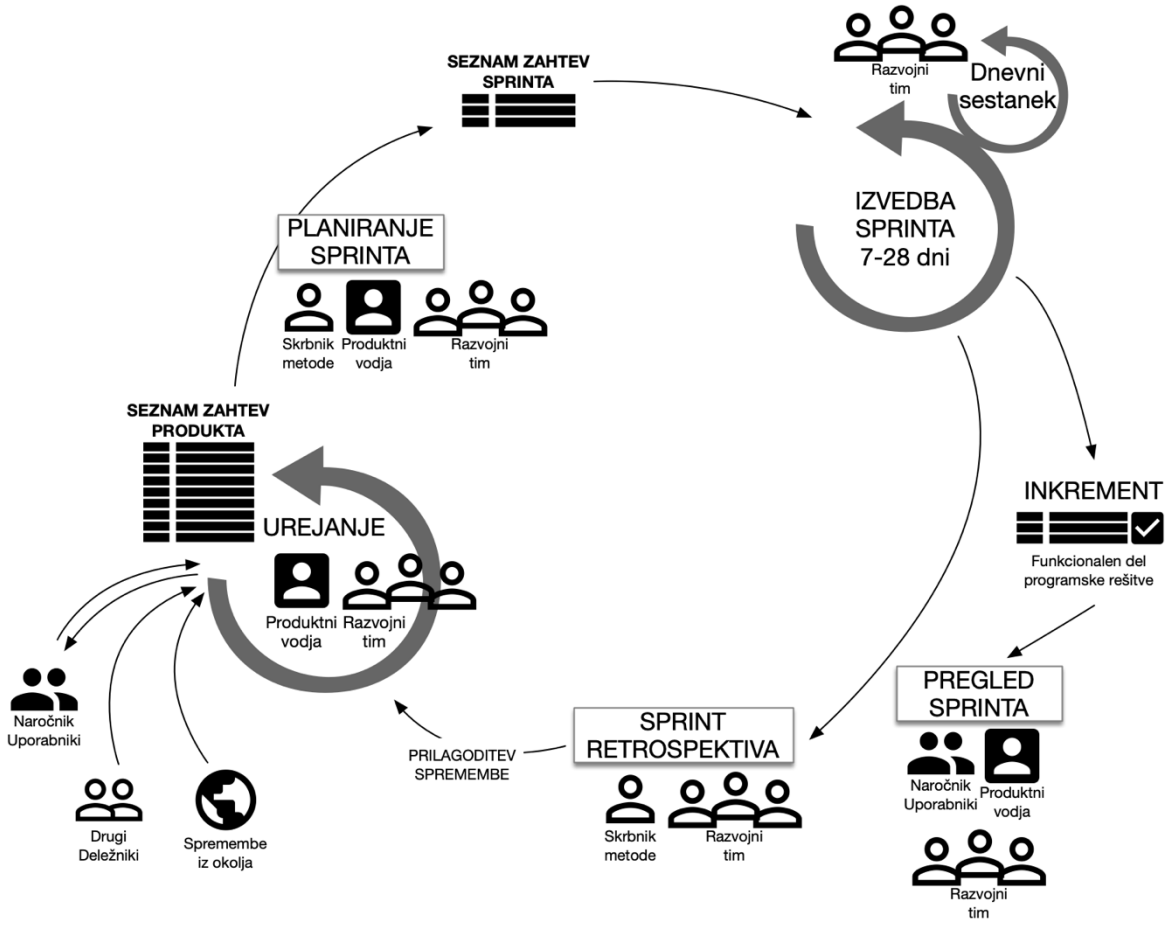
3.2 Aktivnosti/dogodki po metodi Scrum

Aktivnosti po metodi scrum nekateri imenujejo tudi ceremonije, dogodki ali sestanki. Vključujejo planiranje sprinta (angl. sprint planning), kjer se skrbnik metode, produktni vodja in razvojna ekipa sestanejo in izberejo primerne uporabniške zgodbe za vključitev v naslednji sprint. Osnovna aktivnost je nato izvedba sprinta, ki traja od 7 do 28 dni. Tu se razvojni tim zaveže k razvoju pri planiranju sprinta izbranih uporabniških zgodb, ki so vključene v seznam zahtev sprinta. Pomemben sestanek v okviru sprinta so dnevni sestanki (angl. daily scrum) razvojne ekipe. Po zaključenem sprintu pa sta za nadaljnjo optimizacijo tako procesa kot produkta pomembna pregled sprinta (angl. sprint review) in sprint retrospektiva (angl. sprint retrospective). Povratne informacije iz obeh sestankov se upoštevajo pri iterativnem procesu urejanja seznama zahtev produkta in pri optimizaciji dela v prihodnjih sprintih.

Redno se izvaja urejanje seznama zahtev produkta, za katerega je zadolžen produktni vodja. Produktni vodja upošteva povratne informacije tako iz zaključenega sprinta kot od naročnika, drugih deležnikov in okolja. Skupaj z razvojno ekipo ureja in razčlenjuje uporabniške zgodbe in jih tako pripravlja za vključitev v prihodnje sprinte.

Osnovni element metode (angl. scrum artefact) na najnižji ravni je uporabniška zgodba, ki je zahteva spisana iz vidika uporabnika in shranjena v seznamu zahtev produkta. Med elemente metode spadajo še seznam zahtev sprinta (angl. sprint backlog), seznam zahtev izdaje (angl. release backlog) in inkrement (angl.increment). Potek razvoja po metodi Scrum je prikazan na sliki 9.

Slika 9: Diagram razvoja po metodi Scrum



Vir: lastno delo.

3.2.1 Izvajanje sprinta

Metoda Scrum je inkrementalna in iterativna metoda, ki temelji na ponavljajočih se sprintih (krajših ciklih), znotraj katerih se tim zaveže v danem sprintu razviti planirane uporabniške zgodbe oz. funkcionalnosti. Sprinti trajajo 1−4 tedne. Znotraj sprintov se produkt razvija naprej. Rezultat na koncu sprinta je delujoč del programske opreme.

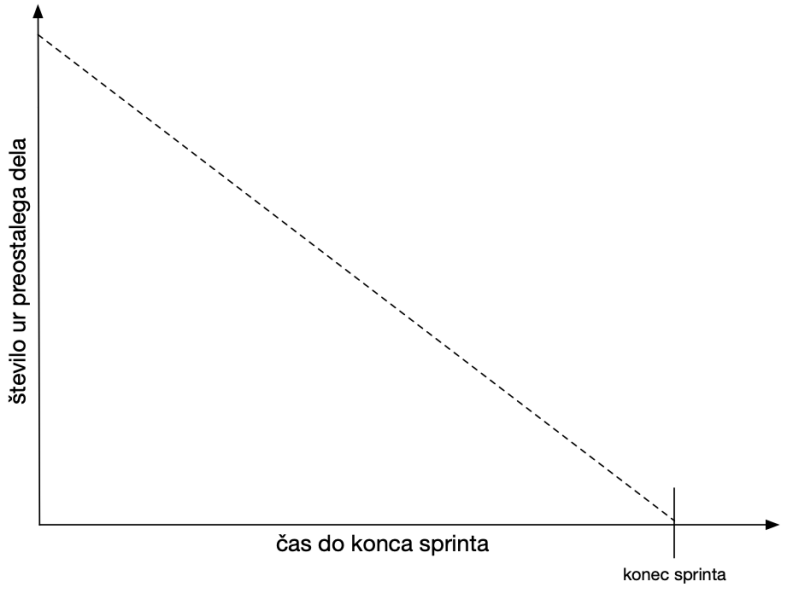
Kratki sprinti omogočajo učinkovitejše prilagajanje spremembam in zaradi pogostega pregleda natančnejšo meritev napredka. S tem se zmanjšuje tudi tveganje, saj bi z daljšimi sprinti ali cikli potencialno dlje časa hodili v napačno smer. Na to se nanaša tudi agilni princip uporabljen pri metodi Scrum, ki spodbuja k hitrim neuspehom (angl. fail fast) in s tem hitrejšemu učenju iz napak.

Osnovni element sprinta je seznam zahtev sprinta, ki vsebuje najpomembnejše uporabniške zgodbe, ki so bile na sestanku planiranja sprinta prenesene iz seznama zahtev produkta. Seznam zahtev sprinta se tekom sprinta ne spreminja, prav tako do konca sprinta niso dovoljene spremembe zahtev ali drugi zunanji vplivi na razvojni tim.

Na vsak dan sprinta se tim sreča na dnevnem sestanku, kjer se vsi člani tima seznanijo z napredkom in dnevnim planom dela. Vsak član tima poroča o delu preteklega dne in delu, ki ga namerava opraviti tisti dan, prav tako izrazi morebitne težave. To je hiter sestanek, ki naj ne bi trajal več kot 15 minut.

Za pregled napredka in napovedovanje bodočega poteka znotraj sprinta se uporablja diagram preostalega dela, ki prikazuje kumulativni preostanek potrebnega dela za dokončanje vseh nalog v seznamu zahtev sprinta glede na čas do konca sprinta. Idealen diagram, kjer so naloge opravljene v predvidenih časovnih okvirih oz. ocenah, je prikazan na sliki 10. Poudariti pa je treba, da je takšna krivulja redka, saj je med osnovnimi principi metode Scrum, da vsega ne moremo predvideti, prav tako pa so viri potrebni za dokončanje uporabniške zgodbe jasno označeni kot ocene (Quigley & Pries, 2011).

Slika 10: Primer idealnega diagrama preostalega dela



Prirejeno po Quigley & Pries (2011).

Diagram preostalega dela prinaša dodano vrednost le, če se ga posodablja pogosto in redno, predvidoma dnevno. Iz njega je hitro razvidno ali razvoj poteka po planu. S tem pregledom se zmanjšuje tveganje, saj je razvidno ali je dnevna učinkovitost (število opravljenih ur), kot je predvidena. V primeru, da je potek pod ali nad planom, so po metodi Scrum možne prilagoditve. Npr. obseg neke funkcionalnosti znotraj uporabniške zgodbe se za optimalni zaključek sprinta lahko poveča ali zmanjša (Quigley & Pries, 2011).

Še ena metrika sprinta je njegova hitrost. Pomeni kumulativne ure dela, ki so na voljo v času trajanja sprinta. Definira se na začetku sprinta in določa h kolikim zahtevam/uporabniškim zgodbam se tim lahko zaveže v sprintu. Hitrost sprinta ni konstanta in se spreminja med projektom.

V enem sprintu se tim zaveže, da bo razvil eno ali več funkcionalnosti oz. t. i. uporabniških zgodb. Ker zgodba predstavlja neko funkcionalnost, opisano iz vidika uporabnika, npr. »Kot uporabnik lahko določam časovni obseg poročila in s tem pridobim pregledne informacije«, je ob koncu vsakega sprinta možno videti delujoč del programske opreme. S tem se med drugim tudi lažje zadovolji potrebo naročnika in drugih deležnikov po vidnih rezultatih med projektom.

3.2.2 Planiranje sprinta

Pred začetkom vsakega sprinta se razvojni tim, produktni vodja in skrbnik metode Scrum srečajo na sestanku planiranja naslednjega sprinta. Cilj sestanka je zaveza udeležencev k razvoju na sestanku izbranih funkcionalnosti/uporabniških zgodb iz seznama zahtev produkta v naslednjem sprintu. Definirajo cilj sprinta, spremembe v seznamu zahtev produkta in ustvarijo seznam zahtev sprinta.

Produktni vodja kot odgovorni za seznam zahtev produkta predstavi zahteve, ki so po pomembnosti in vrednosti najbolj primerne za vključitev v sprint. Razvojni tim oceni obseg in potreben trud za razvoj predstavljenih zahtev ter s tem določi, katere uporabniške zgodbe bodo vključene v sprint. Izbrane uporabniške zgodbe razvojni tim podrobneje pregleda in po potrebi razčleni na manjše uporabniške zgodbe ali naloge ter pripravi plan razvoja inkrementa, ki bo ustrezal zastavljeni definiciji dokončanega (angl. definition of done).

3.2.3 Dnevni sestanki

Po metodi Scrum je pogosta komunikacija ključna. Ena izmed manifestacij le-tega so kratki dnevni sestanki, na katerem se zberejo člani tima in skrbnik metode ter se v 15 minutah pogovorijo o ključnih temah za tisti dan. Cilj je pregled opravljenega dela in napredka proti cilju sprinta ter morebitna prilagoditev seznama zahtev sprinta. Vprašanja, na katera odgovori vsak član tima med sestankom, so:

- Kaj si delal včeraj?
- Kaj boš delal danes?
- Ali ti kaj preprečuje napredek pri delu?

3.2.4 Pregled sprinta

Na sestanku so lahko poleg razvojne ekipe prisotni tako interni kot eksterni deležniki. To je sestanek, na katerem se pregleda in deležnikom projekta predstavi funkcionalnosti razvite v končanem sprintu in jih postavi v kontekst doseganja zastavljenih ciljev. Je za razvojno ekipo priložnost pridobiti povratne informacije in za ostale prisotne priložnost pregleda dela in podajanja mnenja Sestanek ni zgolj enostranska predstavitev, ampak je delovni, kjer se vsi udeleženi pogovorijo o spremembah okolja, okoliščin ali drugih pomembnih vplivih, ter

se dogovorijo o naslednjih korakih in morebitnih spremembah seznama zahtev produkta. Rezultate pregleda sprinta se potem upošteva pri planiranju naslednjega sprinta. Sestanek je časovno omejen na maksimalno 4 ure, vendar je običajno proporcionalen glede na izbrano dolžino sprinta (Rubin & Lichtenberg, 2014).

3.2.5 Retrospektiva sprinta

Po koncu vsakega sprinta se člani razvojnega tima sestanejo s produktno vodjo in skrbnikom metode, da kritično preučijo kvaliteto in učinkovitost sprinta in planirajo izboljšave za prihodnje sprinte. Tim pregleda procese, orodja, komunikacijo, okolje ter diskutirajo in identificirajo tako problematična mesta kot vidike, ki so potekali dobro. Dogovorijo se za potrebne spremembe in aktivnosti za optimizacijo prihodnjih sprintov. Rubin in Lichtenberg (2014) retrospektivo sprinta označita kot enega najpomembnejših praks scrum metode, ki udeležencem daje priložnost prilagoditve konkretnim razmeram in prispeva k oblikovanju prakse kontinuiranih izboljšav.

3.3 Elementi metode Scrum

3.3.1 Uporabniške zgodbe

Začetek vsakega projekta je v definiranju potreb, ki naj bi jih produkt zadovoljil. To tudi pri managementu projektov po metodi Scrum ni drugače, z upoštevanjem želja naročnika se identificira ključne funkcionalnosti se razvijajoče programske opreme. Za identifikacijo potreb imamo na voljo različne strategije. Nekatere med njimi so opisi uporabe produkta po korakih, intervjuji ali anketiranje uporabnikov, opazovanje potencialnih uporabnikov (Quigley & Pries, 2011).

Zahteve so pri metodi Scrum izražene kot uporabniške zgodbe (angl. user stories), ki v svojem bistvu opisujejo zahtevo iz vidika uporabnika oz. konkretne koristi neke funkcionalnosti. Uporabniške zgodbe so napisane čim bolj enostavno in brez podrobnosti. Quigley in Pries (2011) poudarjata, da je jedrnata formulacija bolj učinkovita in je iz vidika zmanjševanja nepotrebnega dela bolj smiselna, saj se zahteve tekom razvoja spreminjajo. Prav tako kratka in jedrnata formulacija spodbuja k pogostejši neposredni komunikaciji med razvojnim timom in produktno vodjo, ki je pred in med razvojem uporabniške zgodbe na voljo za dodatna pojasnila. Cohn (2010) enostavnost formulacije vidi kot dogovor med razvojnim timom in produktno vodjo, da podrobnosti niso potrebne, saj se bodo o podrobnostih pogovorili še pred implementacijo.

Uporabniške zgodbe v nasprotju s tradicionalnimi metodami ne zahtevajo obsežne dokumentacije. Napisane so na kratko, posamezno na papirnatih ali digitalnih karticah, morebitne nejasnosti pa naj bi razrešila direktna komunikacija z naročnikom/uporabnikom in produktno vodjo. Quigley in Pries (2011) pravita, da je neposredna komunikacija

nenadomestljiva, saj skozi razreševanje nejasnosti razvijalcem poda vpogled v dejanske poslovne potrebe naročnika. Cohn (2010) prednosti pogovora pred podrobno dokumentacijo oz. pisnimi dokumenti opredeljuje skozi tri vzorce vedenja:

- Pisni dokument dojemamo bolj formalno in končno kot pogovor. To lahko vodi do tega, da dokument manj kritično pregledamo in nam je pravilnost le-tega samoumevna. Zato je verjetnost, da ujamemo napako v njem, zmanjšana.
- Pri pogovoru dajemo večji pomen pravilnemu in popolnemu razumevanju, ob nejasnostih postavljamo vprašanja.
- Pisni dokumenti zmanjšujejo timsko odgovornost.

Ker je besedilo uporabniške zgodbe opredelitev funkcionalnosti kot zastavljenega cilja, je kljub enostavnosti in kratkosti pomembna formulacija. Tekom celotnega procesa razvoja uporabniške zgodbe je pomembno imeti v mislih uporabnika. Kniberg (2015) zato priporoča, da se pri formulaciji izogibamo tehnični formulaciji, ki bi preusmerila pozornost stran od konkretne naročnikove potrebe. Strukturo uporabniške zgodbe Cohn (2010) opredeli skozi sledečo formulo:

Kot <tip uporabnika> želim <cilj>, da <razlog zakaj>.

Ali na primeru: Kot skrbnik naročnikov želim imeti možnost iskanja kupca po priimku ali številki računa, da lahko pri težavah takoj najdem kupčeve podatke in mu pomagam.

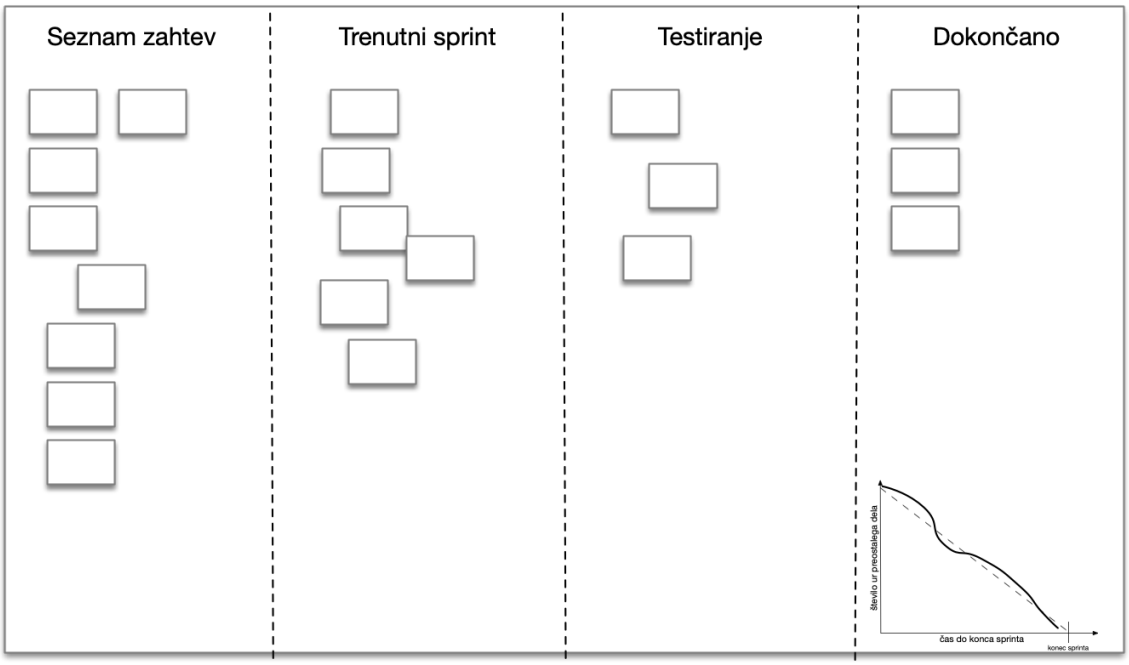
Ena izmed njihovih posebnosti je, da uporabniške zgodbe ne pišejo samo razvijalci in produktni vodja, ampak tudi naročnik sam. Quigley in Pries (2011) v vključevanju uporabnika pri pisanju uporabniških zgodb vidita veliko vrednost, saj ima le-ta poglobljeno znanje iz področja, za katerega se razvija programska oprema.

Določanje uporabniških zgodb pri metodi Scrum pa se od metod zaporednega, planskega razvoja razlikuje tudi po časovni komponenti. Medtem ko se pri plansko vodenih projektih takoj na začetku zbere in čimbolj natančno opredeli vse zahteve za želeni končni produkt, pa metoda Scrum temelji na samoumevnosti sprememb zahtev. Cohn (2010) zagovarja dodelavo zahtev »ravno ob pravem času« (angl. just in time) ter poudarja, da na začetku ni mogoče predvideti vseh zahtev in da je začetni načrt vedno nepopoln. Neizogibno je, da se tekom razvoja pojavijo nove zahteve. Zato ni smiselno natančno definiranje zahtev na začetku, ampak zahteve podrobneje dodelati tekom projekta.

Uporabniške zgodbe so v seznamu zahtev produkta napisane z različno stopnjo natančnosti. Od manjših, dobro razdelanih uporabniških zgodb, pripravljenih za vključitev v sprint, do večjih, obsežnejših, ki jih imenujemo epske uporabniške zgodbe (angl. epic user story). Slednje opredeljujejo zgolj vizijo neke širše funkcionalnosti, niso jasno definirane in zahtevajo med procesom razvoja natančnejšo razčlenitev na manjše uporabniške zgodbe. Za njihov razvoj je potrebnih več sprintov (Cohn 2010).

Ko so uporabniške zgodbe napisane (in morebiti dalje razčlenjene), se poleg hrambe v seznamu zahtev produkta in kasneje sprinta pogosto hranijo na fizični ali virtualni tabli uporabniških zgodb (angl. story board), značilni za metodo Scrum. Kot prikazano na sliki 11, na tabli najdemo stolpce, ki predstavljajo potek od seznama zahtev do zaključka. Uporabniške zgodbe, napisane na kartice, so na tabli uvrščene glede na njihov status. Vsak član tima si lahko izbere poljubno kartico iz seznama zahtev in kartice fleksibilno preureja.

Slika 11: Primer table uporabniških zgodb



Prirejeno po Quigley & Pries (2011).

S tablo uporabniških zgodb je celotnemu razvojnemu timu viden napredek k cilju sprinta in zgodbe, ki sledijo. Odgovornost za urejeno tablo ne leži na enem posamezniku, ampak na celotnem timu. Kot poudarjata Quigley in Pries (2011) skupna odgovornost povečuje sodelovanje članov tima.

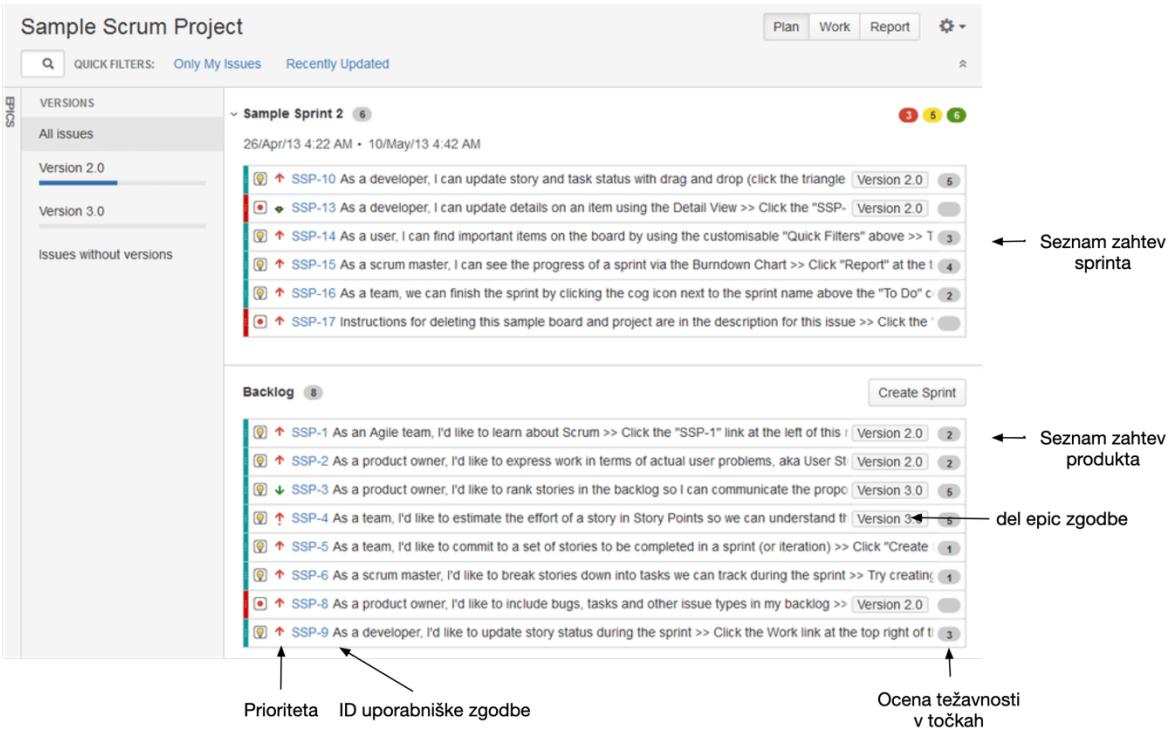
3.3.2 Seznam zahtev produkta

Naročnikove potrebe, prevedene v uporabniške zgodbe, so zbrane v seznamu zahtev produkta, ki vključuje vse v tistem trenutku predvidene funkcionalnosti ter se ga med projektom posodablja in prilagaja spremembam. Vpogled v seznam zahtev produkta imajo vsi člani tima, prav tako lahko uporabniške zgodbe pišejo tako razvijalci, produktni vodja kot naročnik, vendar pa odgovornost za seznam zahtev produkta in spremembe v njem nosi izključno produktni vodja. Ta ga posodablja skozi celoten proces razvoja.

Po Knibergu (2015) seznam zahtev produkta v tabelarni obliki poleg naziva uporabniške zgodbe in ID-ključa vključuje tudi pomembnost ali prioriteto uporabniške zgodbe, oceno

obsega dela, način demonstracije funkcionalnosti in morebitne beležke. S stopnjo prioritete in oceno obsega dela produktni vodja vpliva na to, kaj se prenese v aktualni seznam zahtev sprinta oz. poskrbi, da se najprej razvijejo najpomembnejše funkcionalnosti z največjo vrednostjo za naročnika. Seznam je vedno razvrščen tako, da so najpomembnejše uporabniške zgodbe na vrhu. Na sliki 12 je prikazan primer seznama zahtev produkta v programu Jira, orodju za upravljanje zahtev razvoja informacijskih rešitev.

Slika 12: Seznam zahtev produkta v Jiri



Prirejeno po Atlassian (brez datuma b).

Seznam zahtev produkta vsebuje različno natančno opisane uporabniške zgodbe. Med njimi vsebuje tudi takšne, ki še niso dobro razumljene, so napisane splošno in zahtevajo več kot en sprint za razvoj (epske uporabniške zgodbe). Nahajajo se nižje na seznamu, a se, ko je med projektom na voljo več informacij, le-te razbijejo na manjše uporabniške zgodbe in pomaknejo višje po seznamu. Cohn (2010) kot zadnjo točko dodajanja podrobnosti k uporabniški zgodbi omenja kriterije sprejetja (angl. conditions of satisfaction), s katerimi se definira želeno končno kvaliteto in funkcionalnost uporabniške zgodbe.

Ker produktni vodja sam ne more ocenjevati uporabniških zgodb, to naredi skupaj s timom na sestanku razčlenjevanja uporabniških zgodb (angl. grooming). Prav tako uredijo uporabniške zgodbe po pomembnosti, epske zgodbe razčlenijo na manjše in zgodbe pripravijo za morebitno vključitev v sprint. Cohn (2010) pravi, da je za urejanje seznama zahtev produkta potrebno planirati čas, priporočljivo bi bilo okoli 10 %. Prav tako se mora urejanje seznama zahtev produkta izvajati redno.

Uporabniške zgodbe se med projektom dodajajo (ali odvzemajo) in preko urejanja se spreminjajo tudi prioritete, od katerih je odvisno, ali bo zgodba vključena v naslednji seznam zahtev sprinta ali ne. V izogib temu, da nekatere zgodbe vedno prehitijo nove bolj pomembne naloge, Quigley in Pries (2011) priporočata zviševanje prioritete glede na »starost« naloge. Vsaki nalogi se pripiše datum vključitve na seznam, dlje kot je na seznamu, toliko višja je prioriteta.

Ocena dela v metodi Scrum ni razumljena kot natančna meritev, ampak kot napoved, ki omogoča planiranje sprinta. Cohn (2010) poudarja, da je treba ločevati med oceno in zavezo. Tim se lahko zaveže zgolj ocenam za jasne uporabniške zgodbe na vrhu seznama zahtev produkta, ki so relevantne sedaj in kjer je tveganje napačne ocene manjše. Te so tudi primerne za seznam zahtev sprinta.

Produktni vodja za planiranje potrebuje informacijo o obsegu dela in pričakovano hitrost tima. Pri metodi Scrum enote potrebnega dela niso nujno ure dela, ampak točke (angl. story points), ki predstavljajo kompleksnost uporabniške zgodbe. Obseg in opredelitev ene takšne točke definira tim. Prav tako tim definira, koliko takšnih točk lahko zmorejo znotraj enega sprinta. S tem se pri planiranju sprinta ve, katere in koliko uporabniških zgodb se lahko vključi (Quigley & Pries, 2011; Cohn, 2010).

Poker planiranja (angl. planning poker) je orodje, s katerim celoten tim uporabniškim zgodbam dodeli točke. Skrbnik metode vsakemu članu preda kartice z različnimi ocenami, številkami. Po predstavitvi uporabniške zgodbe in ko so morebitna vprašanja razrešena, vsak član tima izbere ustrezno kartico z oceno. Vsi kartico razkrijejo istočasno. V primeru večjih odstopanj član tima z oceno na enem izmed ekstremov to oceno obrazloži. Možno bi bilo, da je le ta član videl nek aspekt, ki ga ostali niso. Igra spodbuja prosto izražanje mnenj in izogibanje skupinskemu razmišljanju v smislu sledenja drugim (Quigley & Pries, 2011).

3.3.3 Seznam zahtev sprinta

Izbor najpomembnejših uporabniških zgodb se za naslednji sprint izbere in prenese v seznam zahtev sprinta. Tim naj bi se tekom sprinta osredotočil zgolj na zgodbe, vsebovane na tem seznamu in se zavezal k razvoju funkcionalnosti, opisanih v njih v času trajanja sprinta. Quigley in Pries (2011) razložita, da je ta zaveza pogoj za visoko osredotočenost tima k zgodbam in priporočata, da se v seznam zahtev sprinta raje vključi manj zgodb kot tvega neuspeh sprinta. Ob koncu sprinta se lahko pričakuje delujočo programsko opremo z vključenimi novo razvitimi funkcionalnostmi.

3.3.4 Inkrement

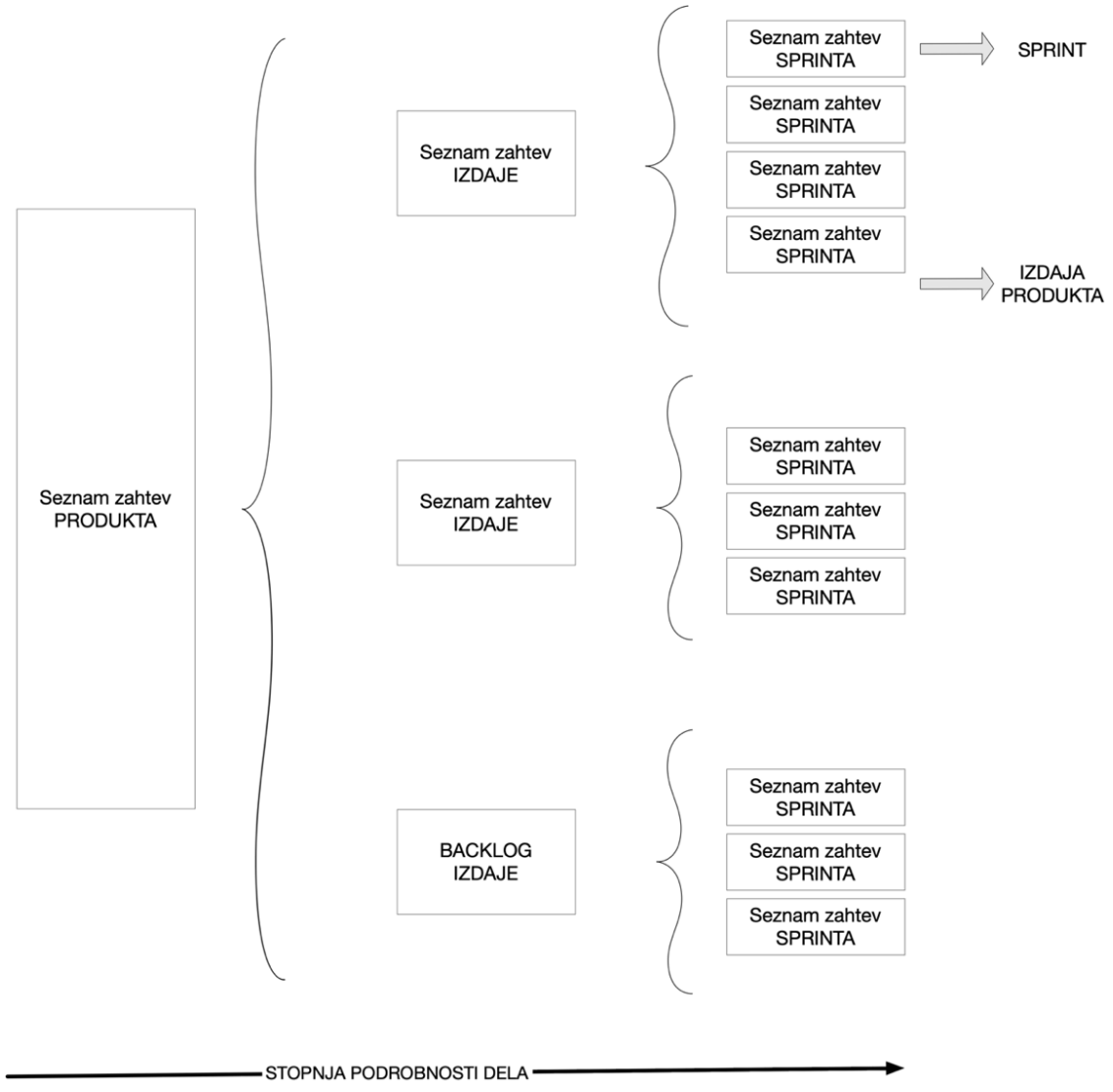
Inkrement je del programske rešitve, ki prispeva k njeni vrednosti in gradi na predhodnjih inkrementih. Znotraj sprinta lahko razvojni tim razvije več inkrementov. Vsak inkrement je

sestavljen iz uporabniških zgodb, ki skupaj sestavljajo uporaben del programske opreme, ki ga je mogoče vključiti v izdajo. Inkrement je definiran z definicijo dokončanega. Definicija dokončanega je pomemben kontrolni seznam, ki vključuje naloge, ki jih mora tim uspešno opraviti preden lahko inkrement označi kot končan. Vključuje med drugim teste, ki preverjajo kvaliteto kode, integracijo v sistem in doseganje kriterijev sprejetja (Rubin & Lichtenberg, 2014).

3.3.5 Seznam zahtev izdaje

Seznam zahtev izdaje vsebuje vse inkremente, ki so primerni za naslednjo izdajo. Običajno so to uporabniške zgodbe oz. funkcionalnosti z najvišjo prioriteto. Na sliki 13 vidimo pregled seznamov zahtev; od seznama zahtev produkta, preko izdaje do seznama zahtev sprinta. Bolj ko se premikamo proti desni, večja je stopnja podrobnosti dela.

Slika 13: Členitev dela na čedalje bolj natančno definirane naloge



Prirejeno po Quigley & Pries (2011).

3.4 Uvajanje (prilagajanje) agilnih metod

Pri uvajanju agilnih metod Fitzgerald, Hartnett & Conboy (2006) navajajo dva pristopa:

- Uvajanje izbrane agilne metode v celoti, saj je za doseganje pozitivnega učinka potrebna prav ta kombinacija praks, poljubna izbira praks ni mogoča.
- Uvajanje elementov različnih agilnih metod glede na potrebe projekta ali okolja.

Pri slednji opciji Fitzgerald, Hartnett in Conboy (2006) omenjajo dva pristopa prilagoditve metode:

- Pristop kontingence dejavnikov (angl. contingency factors) temelji na pripravi nabora različnih metod, iz katerega se nato glede na potrebe in kontekst projekta izbere tisto metodo ali del nje, ki najbolj ustreza. Avtorji dodajajo, da pristop predpostavlja, da imajo člani tima poglobljeno znanje vseh metod v naboru, kar pa ocenjujejo kot manj verjetno.
- Inženiring metode je pristop, po katerem se iz delov različnih metod, ki najbolj ustrezajo posameznim procesom razvoja, sestavi novo prilagojeno metodo in uporabi pri managementu projektov.

Kombiniranje metod zagovarja tudi Cobb (2015) in poudarja, da ne obstajata zgolj dve nasprotujoči si možnosti, ena slapovni in druga agilni pristop k managementu projektov, ampak imamo opravka z lestvico. Na eni strani je bolj tradicionalen planski pristop, kjer je planiranje osredotočeno na čas pred implementacijo, in na drugi strani agilni pristop, kjer se zahteve in plan razvijajo med samo implementacijo projekta. Obe skrajnosti v realnosti težko najdemo, prav tako enak pristop ni primeren za vse projekte.

Eden izmed uspešnejših primerov podjetij, ki so naredile prehod iz tradicionalnih k metodi Scrum, je podjetje Salesforce. Kot je pogosto pri velikih in zrelih podjetij, se je tudi Salesforce boril z zmanjševanjem stopnje rasti, nizko inovativnostjo, zamujenimi roki, zmanjšanjem produktivnosti, povečevanju števila napak in podobno. Uvedbe metode Scrum se je podjetje lotilo radikalno z vseobsežno agilno transformacijo, ne samo na nivoju enega oddelka ali področja, ampak celostno čez celotno podjetje in to v zgolj 3 mesecih. Rezultati so bili odlični: od transformacije leta 2007 do leta 2011 je Salesforce deležnikom prinesel kar 41-% porast letnega povračila investicije (angl. return on investment) (Denning, 2011).

Salesforce se je spremembe lotil celostno, pristop pa je vključeval visoko stopnjo transparentnosti in odprtosti do povratnih informacij iz podjetja. To je omogočalo večjo vpletenost zaposlenih, saj so mnogi lahko aktivno sodelovali pri oblikovanju rešitve. Vodstvo se je odločilo namesto bolj previdne postopne uvedbe metode Scrum za hitro implementacijo na vseh področjih istočasno. S tem so se želeli izogniti morebitni disonanci med oddelki in pokazati svojo odločenost k tej spremembi. Vodstvo je spremembo podpiralo in zanjo prevzemalo odgovornost od začetka uvedbe do konca (Denning, 2011).

Za implementacijo je bil zadolžen multifunkcionalni tim, ki je pripravil načrt celotnega procesa, poskrbel za izobraževanje timov z metodo individualnega usmerjanja (angl. coaching), prirejanja delavnic, odpravljal morebitne ovire na poti k spremembam, spremljal napredek in bil vzor novega načina dela skozi podjetje. Rezultat je bil iterativni način dela z 20-dnevnim ciklom. Fokus je bil na timskih rezultatih in ne na produktivnosti posameznika (Denning, 2011).

3.4.1 Spremembe agilne transformacije

Pri tranziciji k agilnemu managementu projektov so pričakovane in potrebne določene spremembe. Misra, Kumar in Kumar (2011) identificirajo naslednje kot ključne za agilno tranzicijo:

- spremembe organizacijske kulture,
- spremembe v stilu managementa,
- spremembe v strategiji managementa znanja,
- spremembe v procesih razvoja.

3.4.1.1 Organizacijska struktura in kultura

Organizacijska kultura igra veliko vlogo pri spremembi iz plansko usmerjenega managementa projektov k agilnemu. Vključuje stališča, vrednote in vedenja, odnose med ljudmi, način na katerega se sprejemajo odločitve in rešujejo problemi – vse to se mora uskladiti agilnemu pristopu.

Bolj kot obstoječa organizacijska kultura odstopa od agilnih vrednot in praks, težja in daljša bo tranzicija. Spremembe organizacijske kulture so med najtežje izvedljivimi in so zelo počasne, saj to zahteva spremembo zakoreninjenih vzorcev vedenja. Če spremembe nasprotujejo obstoječim normam, bodo te spremembe za posameznike težje. Koch (2005) poudarja, da tega ne smemo podcenjevati in da je za uspeh potreben premišljen in usklajen program utrjevanja želenih vedenj, dokler ta niso ponotranjena.

Pogost izziv so hierarhično strukturirana podjetja z izrazito formalno organizacijsko kulturo in razvijalci z navado samostojnega (samotarskega) dela. Agilne spremembe pa nasprotno gradijo na praksah skupnega timskega dela in odločanja, deljenja znanja in sodelovanja z naročnikom. Neprekinjena, iskrena in odprta komunikacija znotraj tima in z deležniki je v agilni organizacijski kulturi spodbujana. Prav tako se znanje deli med posamezniki in gradi zaupanje.

Pri agilnem managementu projektov veliko vlogo igra odnos z naročnikom, s katero se skupaj sprejema odločitve glede razvoja. Prav tako v nasprotju s tradicionalnim managementom projektov odločanje ni zgolj na strani projektnega managerja, ampak se odgovornost in odločanje razporedi med vse udeležene pri projektu. To zahteva pomemben

premik organizacijske kulture, ki spodbuja zaupanje in spoštovanje (Misra, Kumar & Kumar, 2010).

3.4.1.2 Stil managementa

Uvedba agilnih metod v bolj hierarhično strukturiranih podjetjih zahteva spremembo v managementu. Namesto ukazovanja in kontrole s strani nadrejenih se pri agilnih procesih vodi preko kolaboracije in usmerjanja. Odločitve se sprejemajo skupaj in razvojnemu timu se zaupa, da določene odločitve sprejema tudi sam (Misra, Kumar & Kumar, 2010).

Veliko vlogo igra tudi stopnja usklajenosti stališč in vrednot vodstva z agilnimi praksami. Cao, Mohan, Xu in Ramesh (2009) ugotavljajo, da je za uspeh uvedbe agilnih metod ključna podpora managementa, ki razume prednosti prihajajoče tranzicije. Prav tako mora vodstvo sprejeti in podpirati potrebo po uravnoteženju konvencionalnih organizacijskih procesov in agilnih metod ter s tem preprečiti konflikte, ki bi lahko nastali zaradi neskladja med potrebami vodstva, obstoječe organizacijske kulture in kulture agilnega tima.

3.4.1.3 Spremembe v strukturi managementa znanja

Pri agilnih metodah se odmika od praks ustvarjanja obsežnih dokumentacij in stremi k spontanim in fleksibilnim metodam managementa znanja. Znanje je v agilnem timu neformalno in pogosto »shranjeno« v glavah članov. Zato je znotraj tima pomembno tesno sodelovanje in medsebojno deljenje znanj (Misra, Kumar & Kumar, 2010).

3.4.1.4 Spremembe v procesu razvoja

V agilnih timih je pomen procesov in sledenja le-tem manjši. Opira se na agilne smernice in ne določa rigidnega poteka korakov. Procesi razvoja se spremenijo iz zaporednih v iterativne korake ali cikle; od sledenja planu k razvoju z več nepredvidljivosti in s sprejemanjem sprememb.

Misra, Kumar in Kumar (2010) v kvantitativni študiji kot najpomembnejše spremembe razvojnega procesa pri tranziciji k agilnim metodam navajajo:

- proces razvoja osnovan na ljudeh, iteracijah, rednem testiranju kode,
- evolucijski in iterativni razvoj funkcij produkta,
- razvoj, toleranten do sprememb,
- sprejemanje nejasnosti v razvoju.

3.4.2 Faktorji uspeha pri uvajanju metode Scrum

Vsaka začeta sprememba ni nujno uspešna. Čeprav agilne metode dokazano prinašajo pozitivne učinke pri managementu projektov, pa obstajajo faktorji, ki pozitivno ali negativno vplivajo na uspešnost uvedbe metode Scrum in drugih agilnih metod. Avtorji (Cao, Mohan, Xu & Ramesh, 2009; Ozierańska, Skomra & Dorota, 2016; Misra, Kumar & Kumar, 2010) opisujejo različne faktorje, med njimi velikost projekta, organizacijsko strukturo in kulturo, poslovne procese, tehnologijo, način prenosa znanja in človeške faktorje, kot so vrednote in vodstveni stil ter samo poznavanje agilnih metod. Pri uvajanju metode Scrum Ozierańska, Skomra in Dorota (2016) identificirajo 5 kategorij faktorjev, ki vplivajo na uspešno uvedbo metode: projektni tim, psihološki in kulturni vidiki, proces in metoda, okolje ter tehnologija.

3.4.2.1 Projektni tim

V to kategorijo Ozierańska, Skomra in Dorota (2016) združijo otipljive in neotipljive faktorje povezane s timom. Prostorski faktorji kot fizična prisotnost na isti lokaciji pozitivno vplivajo na uvedbo, saj ustvarjajo ugodne pogoje za razvoj med seboj povezanega tima. Posamezniki lažje komunicirajo, se usklajujejo in učijo drug od drugega. Lažje se razvije timski duh. Delo od doma oz. na daljavo pa na uspešnost uvedbe iz zgoraj opisanih razlogov vpliva negativno. Majhen tim pozitivno vpliva na uspešnost uvedbe, saj se tako lažje razvijejo pozitivni odnosi med člani. Izboljšana komunikacija, prenos informacij, zaupanje, identifikacija s timom so prednosti, ki so pomembne za razvoj samoorganiziranega tima.

Negativno pa na uspešnost uvedbe metode Scrum vpliva, če imajo člani tima ločena področja kompetenc, npr. vsak na nekem ožjem tehničnem področju. Po metodi Scrum odgovornost za projekt leži na celotnem timu, fokus na zgolj lastno delo zaradi ozke specializacije pa zmanjšuje kolektivni občutek odgovornosti celotnemu projektu. Prav tako negativno vpliva delo posameznikov v različnih timih oz. na različnih projektih. To zmanjšuje občutek pripadnosti timu in odgovornosti za posamezen projekt (Ozierańska, Skomra & Dorota, 2016).

Podkategorija cilji tima ima samo en faktor, ki opredeljuje morebitno neskladje komuniciranih ciljev na eni strani in realnostjo dejanskih nalog na drugi. Odstopanja vodijo v nemotiviranost in frustracije (Ozierańska, Skomra & Dorota, 2016).

3.4.2.2 Psihološki in kulturni vidiki

Kategorija se nanaša na neotipljive faktorje povezane z ljudmi v timu. Lastnosti članov tima, kot sta npr. individualizem in nedisciplina pri udeleževanju sestankov, negativno vplivajo na implementacijo metode Scrum. Prav tako zmanjšuje produktivnost tima, če so člani v svetovalni vlogi aktivni drugje ali pa če jih je strah sodbe drugih izven tima. To vodi v zmanjšano produktivnost in zaradi zadržkov deljenja informacij do zmanjšanega pregleda

nad napredkom projekta. Negativno vpliva tudi, če posamezniki ne ocenjujejo realno nalog in zmožnosti za njihovo izvedbo. To vodi v preveč dela, utrujenost, manjšo motiviranost in posledično slabo vpletenost v projekt in zmanjšanje motiviranosti (Ozierańska, Skomra & Dorota, 2016).

Na drugi strani pa visoke kompetence, visoka motiviranost in odprtost članov tima ter prisotnost prijateljskih odnosov med člani, pozitivno vplivajo na uspešnost uvedbe metode Scrum (Ozierańska, Skomra & Dorota, 2016).

3.4.2.3 Proces in metoda

Pod to kategorijo Ozierańska, Skomra in Dorota (2016) identificirajo faktorje povezane z izvajanjem procesov po metodi Scrum. Kot pomemben faktor se izkaže ustrezno delo skrbnika metode in produktnega vodja. Predvsem skrbnik metode igra veliko vlogo pri zagotavljanju optimalnih pogojev za delo razvojnega tima, če le-ta ni prisoten na sestankih ali nameni vlogi zgolj omejeno truda in časa, to negativno vpliva na uspešnost uvedbe metode Scrum.

Vpliv na uvedbo ima tudi predhodno poznavanje agilnih metod med člani tima in ostalimi deležniki. Več znanja, izkušenj in poznavanja agilnih praks ter pozitivna stališča do le-teh pozitivno vplivajo na uvedbo agilnih metod (Cao, Mohan, Xu & Ramesh, 2009).

3.4.2.4 Okolje

Tim v okviru metode Scrum deluje znotraj ekosistema podjetja, kjer procesi niso nujno usklajeni s procesi te metode . Pomemben kriterij je, da vodstvo in tudi preostalo podjetje vidijo vrednost v novi metodi dela. Ozierańska, Skomra in Dorota (2016) poudarjajo, da morebitna organizacija sestankov v času dogodkov po metodi Scrum ali rigidnost planskega projektnega managementa na nivoju podjetja zmanjšuje možnosti delovanja tima po principih metode Scrum. V primeru, da več timov dela na med seboj odvisnih delih projekta, postanejo pomembni načini komunikacije potrebni za sinhronizacijo.

3.4.2.5 Tehnologija

Kriteriji povezani s tehnologijo se nanašajo na pogostost tehničnih težav in kako hitro jih je tim sposoben rešiti, saj to vpliva na zmožnost tima, da dela kontinuirano. Negativno vpliva tudi, če tim ne pozna tehnologij, ki jih potrebuje za implementacijo. Oba primera privedeta do morebitnih napačnih ocen, zamud in prenosa zgodb v naslednji sprint (Ozierańska, Skomra & Dorota, 2016).

4 PRIMER UVEDBE METODE SCRUM V PODJETJU

Metoda Scrum je ena izmed najbolj razširjenih metod projektnega managementa pri razvoju programske opreme. Je skupek smernic, ki predvsem preko ponavljajočih se ciklov dela in odprtosti do sprememb omogočajo večjo fleksibilnost v procesu razvoja in posledično boljše poslovne rezultate. Ampak ali je metoda Scrum v svoji celoti primerna za vsako podjetje ali projekt? Privede v vsakem primeru do povečanja učinkovitosti pri delu in uspešnosti projektov?

V magistrskem delu želim na primeru podjetja raziskati uporabo te metode za povečanje uspešnosti pri projektih razvoja programske opreme ter način uvedbe metode. Cilj raziskave je potrditev koristi, ki jih naj bi po teoriji prinesla uvedba in uporaba metode Scrum, ter boljše razumevanje metode Scrum v praksi.

4.1 Raziskovalna vprašanja

4.1.1 Uspešnost projektov

Iz teoretičnega dela magistrske naloge sledi, da metoda Scrum podjetjem prinaša večjo učinkovitost managementa projektov in bistveno poveča možnosti uspeha projekta. Chaos študija uspešnosti projektov med leti 2011 in 2015 beleži neuspeh pri zgolj 9 % projektov vodenih po agilni metodi in kar 29 % projektov vodenih po slapovni metodi (Standish Group International, Inc., 2015). Odstotek uspešnih projektov je pri agilnih metodah skoraj štirikrat višji kot pri slapovni metodi. Iz tega lahko sklepamo, da se podjetja lotevajo uvedbe metode Scrum z namenom izboljšanja uspešnosti projektov. Vendar kako je s tem v praksi? V tematskem sklopu uspešnosti projektov vodenih po metodi Scrum zastavljam naslednja raziskovalna vprašanja:

- So v praksi projekti razvoja programske opreme uspešnejši po uvedbi metode Scrum?
- So uspešnejši po vseh ali samo nekaterih kriterijih uspešnosti?

Kriterije za opredelitev uspešnosti projekta osnujem na Gollner in Baumane-Vitolina (2016) modelu uspešnosti ERP-projekta. Avtorja definirata pet sklopov kriterijev: projektni management, čas in proračun, kvaliteta ERP-sistema, ekonomska vrednost in zadovoljstvo uporabnikov. Za pridobitev odgovora na raziskovalni vprašanji uporabim spodnje kriterije.

Kriterij 1: Z uvedbo metode Scrum je kvaliteta projektnega managementa izboljšana.

Kriterij 2: Z uvedbo metode Scrum je projekt razvoja programske opreme učinkovitejši. To pomeni hitrejši razvoj programske opreme z nižjimi stroški.

Kriterij 3: Programske rešitve po uvedbi metode Scrum so ocenjene kot bolj kvalitetne.

Kriterij 4: Z uvedbo metode Scrum se izboljša zadovoljstvo naročnikov.

Kriterij 5: Oceno ekonomske vrednosti lahko poda naročnik in ne ponudnik programske opreme oz. bi to presegalo okvire intervjuja. Zato ta kriterij izpuščam.

4.1.2 Uvedba metode Scrum

Z uvedbo metode Scrum si podjetja obetajo višjo učinkovitost, izboljšano inovativnost, boljše poslovne rezultate ter zadovoljstvo uporabnikov in ostalih deležnikov. Pomemben pa je tudi način uvedbe. Iz teorije vemo, da določeni faktorji vplivajo na uspešnost uvedbe. Prav tako Cobb (2015) priporoča, da se pri uvedbi upošteva specifike podjetja in projektov ter kjer potrebno metodo prilagodi. V tem kontekstu zastavljam naslednja raziskovalna vprašanja:

- Ali je za učinkovito uporabo metode Scrum potrebno metodo prilagoditi konkretnim potrebam podjetja v praksi?
- Kakšne so bile prilagoditve metode Scrum lastnim potrebam?

Poleg odgovorov na zgornji raziskovalni vprašanji želim raziskovati tudi način uvedbe metode Scrum, in sicer želim izvedeti:

- kakšen je bil potek uvedbe,
- kako se je razdeljevalo vloge predvidene po metodi Scrum
- s kakšnimi ovirami se je podjetje srečevalo, kje so se pojavile težave pri uvedbi, katere so bile predvidene in katere nepredvidene,
- je bila metoda Scrum uvedena v celoti ali v kombinaciji z drugimi metodami in zakaj.

4.2 Metodologija

Za pridobitev vpogleda v projektni management po metodi Scrum v praksi sem se odločila za kvalitativno metodo raziskovanja: polstrukturirani intervju. Pristop je poizvedovalne narave, saj sem želela ne samo odgovoriti na raziskovalna vprašanja, ampak zajeti morebitne dodatne nepričakovane informacije povezane z uporabo metode Scrum v praksi.

Pri poglobljenih intervjujih se v nasprotju s kvantitativnimi metodami izbere manjši, a skrbno izbran vzorec. To je lahko tudi zgolj ena oseba, ki je strokovnjak na področju, ki ga raziskujemo (Walle, 2015). Blaž Strle, izbrani intervjuvanec, je certificirani skrbnik metode z večkratnimi izkušnjami uvedbe metode Scrum v podjetje. Prav tako je v konkretnem primeru poleg odgovornosti za uspešno uvedbo metode Scrum opravljal tudi vlogo skrbnika metode Scrum in imel tako celosten pregled nad procesom uvedbe.

Vprašalnik polstrukturiranega intervjuja temelji na podatkih iz strokovne literature na temo metode Scrum. Smotrnost vprašanj sem najprej preverila pri osebi, ki metodo Scrum

uporablja pri vsakodnevnem delu kot vodja razvojnega tima. Po popravku vprašalnika sem izvedla poglobljen intervju z Blažem Strletom, odgovorno osebo za uvedbo metode Scrum v oddelku. Vnaprej definirana vprašanja so služila kot smernice pogovora in so bila odprte narave. Omogočala so, da sem na eni strani pridobila osnovne informacije oz. odgovore na vnaprej definirana vprašanja in na drugi strani omogočala pogovor, spontano postavljanje vprašanj in pridobivanje globljega vpogleda v temo. Govorila sva splošno, na nivoju vseh projektov. Podrobnosti konkretnih projektov in naročnikov iz razlogov varovanja poslovnih podatkov niso vključeni.

4.3 Podjetje in produkt

Adacta d.o.o. (danes Be-terna) je Microsoftov partner in naročnikom ponuja implementacijo portfelja Microsoft 365 ERP programskih rešitev in rešitev za upravljanje odnosov s strankami (angl. Customer Relationship Management, v nadaljevanju CRM). Metoda Scrum je bila implementirana v ERP Dynamics AX oddelek podjetja. Oddelek je zadolžen za implementacije ERP-programske rešitve Microsoft Dynamics AX, ki jo danes poznamo kot Dynamics 365 Finance and supply chain. Je del MS Dynamics 365 paketa rešitev za srednje velika do velika podjetja in omogoča brezhibno integracijo s preostalimi aplikacijami iz Dynamics družine.

To je gotova (angl. off-the-shelf) rešitev, ki je fleksibilna in se jo z nastavitvami in dodelavami prilagodi procesom in integrira z obstoječimi sistemi naročnika. Rešitev podjetjem omogoča boljšo organizacijo, avtomatizacijo in optimizacijo procesov, digitalizacijo podatkov in doseganje digitalne transformacije, ki je danes praktično nujna za nadaljnje uspešno poslovanje. Osnovna rešitev vključuje funkcionalnosti na področju upravljanja financ, proizvodnje, dobavne verige, upravljanja sredstev, management projektov in kadrovske službe (Be-terna, brez datuma).

4.4 Uvedba metode Scrum

Metoda Scrum se je leta 2014 uvajala v delo oddelka Microsoft Dynamics AX v podjetju Adacta d.o.o. To je bil že drugi oddelek v podjetju, ki je pričel z delom po tej metodi. Pobudnik uvedbe je bilo samo vodstvo podjetja, ki ga vidimo kot sponzor projekta. Za management spremembe pa je bil zadolžen Blaž Strle (certificirani skrbnik metode Scrum), takrat v vlogi direktorja Microsoft Dynamics CRM-oddelka, v katerega je metodo Scrum že uspešno uvedel. Njegova vloga pri uvedbi je bila poleg vseh aktivnosti managementa same uvedbe metode Scrum tudi aktivno opravljanje vloge skrbnika metode za vseh 5 timov.

Pred metodo Scrum se je v oddelku uporabljala tradicionalna slapovna metoda, ki ni bila več zadostno učinkovita. Cilj uvedbe metode Scrum je bil v enostavnejšem doseganju zadanega nivoja kvalitete implementirane rešitve, boljšega pokrivanja pričakovanj naročnika in izboljšana obvladljivost projektov. Pred uvedbo so se soočali s poznim oz. prepoznim

prepoznavanjem napak. Napake ali celo napačno zastavljene predpostavke so bile identificirane šele ob koncu razvojnega cikla. Razlog za takšno stanje je bil pogosto v testiranju, ki se je v večji meri izvajalo zgolj na koncu razvojnega cikla. Pogosto je bilo potrebno projektom dodati mesece dela in to v trenutku, ko bi projekt morali že zaključevati. Strle razloži, da so bili projekti bistveno manj obvladljivi.

Nasprotno so si z inkrementalim razvojem manjših kosov in sprotnim testiranjem obetali boljšo obvladljivost projektov. K temu lahko dodamo, da po metodi Scrum pričakujemo povratne informacije naročnika skozi celoten proces razvoja, kar pomeni boljšo skladnost končne rešitve s pričakovanji. Uvedba je trajala okoli 2 meseca, potem pa so bili potrebni nadaljnji 4 meseci, da se je sprememba na metodo Scrum ukoreninila v vsakodnevne procese in aktivnosti.

Metoda Scrum kot iterativni način dela je bil uveden v življenjskem ciklu razvoja produkta samo v fazi izgradnje/razvoja. Strle inkrementalno metodo ocenjuje kot neprimerno za celoten razvojni cikel, kadar gre za implementacijo gotovih rešitev, kot je MS Dynamics AX. Stroški in roki so pogodbeno določeni, zato odmik večinoma ni mogoč. Večja odstopanja od plana tako niso smotrna ali zaželena.

Pred implementacijo/razvojem delo poteka po planski metodi. Strle razloži, da je tveganje za razvoj v napačno smer brez vsaj okvirnega načrtovanja in natančne opredelitve ciljev na začetku prevelik. Z inkrementalnim razvojem in osredotočanjem zgolj na naslednjih 14 dni se lahko izgubi celotna slika želenega rezultata. Na začetku razvojnega cikla je faza intenzivnejše analize in planiranja, kar je v nasprotju z metodo Scrum, ki zagovarja definiranje zgolj osnovne vizije produkta z namenom čim hitrejšega začetka razvoja in izdaje delujočega dela programske opreme. Kasneje se iz analize procesov in natančneje definiranih zahtev oblikujejo uporabniške zgodbe in shranijo v seznam zahtev produkta. Uporabniške zgodbe se tekom razvoja podrobneje dodelajo.

4.4.1 Faze uvedbe metode Scrum

4.4.1.1 Spoznavanje z metodo Scrum

Večina zaposlenih metode Scrum ni poznala, zato je bilo v začetni fazi pomembno zaposlene izobraziti. Za vsak tim so bile organizirane delavnice, kjer je Strle predstavil tako teorijo metode Scrum kot praktične primere uporabe iz CRM-oddelka. Člani timov so imeli možnost postavljanja vprašanj. Na voljo so jim bile poleg literature tudi razne predloge, npr. predloga za pisanje uporabniške zgodbe in kontrolni seznam definicije dokončanega.

4.4.1.2 Opredelitev vlog po metodi Scrum in strukture timov

V naslednji fazi se je razdelilo vlogi produktnega vodje in skrbnika metode ter definiralo time. Vlogo skrbnika metode je na začetku za vse time prevzel Strle osebno. Kasneje pa so prišli do ugotovitve, da en skrbnik metode za vse time ne zadošča in so znotraj timov identificirali najbolj primerno osebo za to vlogo. Pretežno je bila ta naloga dodeljena vodji tima. Vlogo produktnega vodja so pretežno prevzeli projektni managerji. Ker so projektni managerji že prej s timom sodelovali večinoma kolaborativno in ne direktivno, ni bilo velikih razlik v pristojnostih ali hierarhijah.

Razvojni timi so po uvedbi metode Scrum ostali nespremenjeni. Že prej so bilo timi sestavljeni multifunkcijsko, torej iz posameznikov iz različnih področij in tako je tudi ostalo po uvedbi, le da so določene osebe dodatno opravljale še vlogo skrbnika metode ali izjemoma produktnega vodje. Tim je sestavljalo od 5 do 8 posameznikov, po funkciji so bili svetovalci, razvijalci, testerji in vodje timov.

4.4.1.3 Priprave in uvedba aktivnosti po metodi Scrum

Dogodke oz. aktivnosti po metodi Scrum so udeleženci procesa pred pričetkom kot ponavljajoče dogodke prejeli direktno v koledar. S tem se je implicitno nakazalo visoko pomembnost sestankov, kot so dnevni sestanek, sestanek razčlenjevanja zgodb ali retrospektiva. Pričakovana je bila brezpogojna udeležba. Uveljavitev rutine še posebej pri dnevnih sestankih Strle ocenjuje kot ključno za uspešno uvedbo metode Scrum. Tudi Ozierańska, Skomra in Dorota (2016) poročajo o velikem pomenu redne udeležbe in točnosti na sestankih. Neudeležba posameznikov vodi v splošen negativen odnos do sestankov.

Teden pred uradnim začetkom so bile planirane priprave na uvajanje nove metode in so vključevale tako tehnično pripravo kot pripravo udeležencev procesa. Vključene so bile naslednje aktivnosti:

- priprava Scrum table
- priprava programske opreme Jira za upravljanje seznama zahtev produkta in seznama zahtev sprinta,
- priprava in pošiljanje zahtev za dogodke po metodi Scrum direktno v koledar udeležencev.
- udeleženci se sestanejo že v pripravljalnem tednu na dnevnih sestankih,
- prvi sestanek razčlenjevanja zgodb za kasnejši prvi sestanek planiranja sprinta,
- sestanek svetovalcev in razvijalcev namenjen razreševanju zadnjih nejasnosti,
- vsi udeleženci so imeli petek pred tednom začetka rezerviran za individualne priprave na metodo Scrum.

4.4.1.4 Začetek dela po metodi Scrum

Začetek dela po metodi Scrum je bil časovno natančno določen z datumom 31. 3., vse priprave so morale biti do takrat zaključene. Prvih nekaj sprintov je spremljal tudi zunanji mentor za metodo Scrum (angl. Scrum coach), ki je time podpiral, opozarjal na morebitne izboljšave v procesu in bil na voljo za vprašanja. Dnevne sestanke in uveljavitev te rutine Strle ocenjuje kot ključno. V fazi uvedbe se je kot skrbnik metode udeleževal dnevnih sestankov vseh timov, ki so bili zato ustrezno časovno zamaknjeni.

4.4.1.5 Aktivno vzdrževanje stanja

Ker vsaka sprememba potrebuje nekaj časa, da preide v navado ali splošno prakso, je bilo tudi tukaj potrebno aktivno spodbujanje k vzdrževanju dogodkov in procesov po metodi Scrum. Potrebno je bilo dodatno informiranje in zagotavljanje spoštovanja procesa in principov po metodi Scrum pri vseh udeleženih, vključno z vodstvom.

Strle je v vlogi skrbnika metode skrbel za korektno izvajanje metode Scrum in skrbel, da je imel tim najboljše pogoje za doseganje zadanih ciljev. Poudarja, da je bilo ključnega pomena, da je tim ščitil pred vplivi in interesi posameznikov izven tima, tudi posameznikov na vodstvenih pozicijah, vseh, ki so bili do sedaj navajeni vnašati naloge in spremembe v tim kadarkoli. Z metodo Scrum je tim postal zaščiten pred takšnimi vplivi, tako da se je lahko osredotočil na naloge, opredeljene v seznamu zahtev sprinta.

4.4.1.6 Prilagajanje metode, kjer potrebno

Tekom managementa projektov po novi metodi se je ugotovilo, da je na določenih mestih potrebna prilagoditev:

- Vsak tim potrebuje svojega skrbnika metode.
- Metodo Scrum ni možno izvajati pri nalogah popravkov po prehodu v živo.
- Struktura tima je ostala nespremenjena, čeprav so preizkusili tudi opcijo homogenih timov (člani imajo enake funkcije), ki pa se je izkazala za manj učinkovito.
- Opis uporabniških zgodb iz vidika funkcij in ne nujno zgolj uporabnika.
- Združevanje vlog skrbnika metode, vodje tima in produktnega vodja.

4.4.2 Rezultati uvedbe metode Scrum

4.4.2.1 Samoorganizirani razvojni timi

Metoda Scrum je bila v roku 2 mesecev uspešno uvedena pri 5 od 6 timov. To pomeni, da so timi sodelovali pri dogodkih, upravljali z elementi po metodi Scrum in bili

samoorganizirani. Prav tako je bilo spodbujano deljenje znanja in skupno prevzemanje odgovornosti.

Razlog za neuspeh 6. tima je bil v podporni fazi projekta (angl. hypercare). Ta pomeni, da je implementirana rešitev že v živo (angl. go-live) in se rešujejo zadnje napake. Kadar v tej fazi pride do problemov, ti zahtevajo takojšnjo reakcijo, saj je sistem že v uporabi. Planiranje popravkov po sprintih tako ni mogoče. 6. tim je kasneje ob prevzemu novega projekta prav tako uspešno uvedel metodo Scrum.

Ozierańska, Skomra in Dorota (2016) identificirajo ključne faktorje za razvoj učinkovitega tima po metodi Scrum. Če faktorje v tabeli 4 primerjamo s stanjem v oddelku vidimo, da so vsi faktorji uspeha uvedbe popolnoma ali delno izpolnjeni. Uspešna uvedba metode Scrum na nivoju tima sovpada s tem modelom.

Tabela 4: Faktorji uspešnosti uvedbe metode Scrum v kategoriji tim

Podkategorija	Faktor	Smer vpliva na uspešnost uvedbe	Uvedba v oddelku
Prostorski faktorji	Delo v fizični bližini.	+	Ja, člani tima so fizično v eni ali v bližnji pisarni.
	Možnost dela na daljavo.	-	Možnost za delo od doma sicer obstaja, ampak je omejena. Velika večina vse dneve v tednu dela v pisarni. To se je kasneje leta 2020 zaradi COVID-19 ukrepov spremenilo.
Sestava tima	Majhno število članov tima.	+	Ja, v timu je 5−8 članov.
	Različna kompetenčna področja članov tima.	-	Delno. V timu so različni profili, vendar je zaželeno sodelovanje in deljenje znanja. Razlog za heterogeno strukturo znotraj tima je ravno zmanjšanje individualizma ter razvoj skupinskega dela in skupnih odgovornosti.
	Deljena razpoložljivost članov tima.	-	Ne, člani tima so pretežno 100 % zavezani projektu njihovega tima. Če se specifična znanja člana tima potrebuje drugje, se naloga prenese v njegov tim.
Cilji tima	Neskladje v odgovornostih in odstopanja od prvotno zastavljenih ciljev.	-	Ne, člani tima so odgovorni za razvoj definiranih uporabniških zgodb; zunanji vpliv, dodajanje novih nalog ali sprememb prioritet nalog znotraj trajanja sprinta ni dovoljeno.

Prirejeno po Ozierańska, Skomra & Dorota (2016).

Ključne pozitivne učinke uvedbe metode Scrum na nivoju tima Strle vidi v boljši strukturi dela, ki jim omogoča enotnejše in preglednejše delo. Predvsem se lahko tim osredotoči na definirane naloge in dela skupaj v eno smer.

4.4.2.2 Struktura tima

Pri strukturi tima so v oddelku preizkusili dva modela:

- Homogene strukture tima, kjer so razvijalci in svetovalci v različnih timih.
- Heterogene strukture tima, kjer je v timu kombinacija razvijalcev in svetovalcev.

Strle prednost homogenih struktur pred heterogenimi vidi v učinku večjega strokovnega razvoja posameznikov. Znotraj homogenega tima je učenje in prenos novih znanj specifičnega področja omogočeno s samim delom in interakcijo z drugimi strokovnjaki istega področja. Na drugi strani pa je prednost heterogenih timov razvoj pripadnosti in zavezanosti skupnemu cilju. Tim ima skupno odgovornost za dokončanje projekta, kar jih motivira in žene naprej.

Na koncu so se prav zaradi tega človeškega faktorja − skupinske povezanosti in odgovornosti − odločili za (obstoječe) heterogene time, sestavljene tako iz svetovalcev kot razvijalcev. Strle vseeno poudarja, da je bilo pri odločitvi potrebno tehtati prednosti ene in druge strukture. Brez upoštevanja teh psiholoških faktorjev, ki vplivajo na skupinsko učinkovitost, bi bile homogene strukture bolj učinkovite, saj se strokovnjaki lahko osredotočijo zgolj na tisto delo, za katerega so edinstveno usposobljeni, in na tista področja, pri katerih imajo poglobljeno znanje. Kljub takšni optimizaciji dela pa je v praksi motiviranost posameznikov manjša, če ne vidijo neposrednega vpliva njihovega dela na delo drugih, npr. posledic zamujenega ali slabo opravljenega dela. Skupna učinkovitost brez skupinskega dela, učinkovite komunikacije in deljenja znanja upade. To sovpada s tezo Ozierańske, Skomre in Dorote (2016), da individualizem posameznikov kot nasprotje skupinskemu delu, negativno vpliva na uvedbo metode Scrum.

Stalni timi lahko bolje razvijejo skupinski duh in gradijo na učinkovitosti. Zato člani timov pretežno ostajajo znotraj definiranih timov za ves čas trajanja projekta, aktivni so vedno na enem projektu in ne na več projektih simultano. Razvojni timi so majhni, sestavljeni iz 5−8 posameznikov. Iz tega sledi, da vsak tim nima vseh možnih specifičnih znanj. Če uporabniška zgodba zahteva znanje iz področja, ki ga nima noben član tima, se namesto premestitve strokovnjaka iz drugega tima, dodeli zgodbo timu, ki to znanje ima. Strle razloži, da je za rešitev pomembno poznavanje področja uporabniške zgodbe in ne celotnega projekta. Ta način v oddelku dobro deluje, saj kljub temu, da timi delajo na različnih projektih, se pri vseh implementira ista osnovna ERP-rešitev.

4.4.2.3 Organizacijske spremembe

Projektni managerji so po uvedbi metode Scrum prevzeli vlogo produktnega vodja. Ker so že prej s timom delali kolaborativno in večinoma niso zavzemali za projektnega managerja značilne direktivne vloge, je bila tranzicija tekoča in ni vključevala bistvenih sprememb. Vloga produktnega vodja je v prvi vrsti komunikacija z naročnikom in ne upravljanje z razvojnim timom poudari Strle. Odgovorni so za pregled kosov programske opreme z naročnikom, za usklajevanje zahtev, upravljanje s seznamom zahtev produkta in priprave uporabniških zgodb za vključitev v seznam zahtev sprinta skupaj z razvojnim timom.

Struktura timov je po uvedbi metode Scrum ostala nespremenjena, vključno z obstoječimi hierarhijami. Vsak tim ima vodjo, ki v večini primerov opravlja tudi vlogo skrbnika metode. Razlog za to dvojno vlogo je optimizacija, saj bi bil skrbnik metode v polnem delovnem času potrata človeških virov razloži Strle. V določenih primerih, npr. pri manjših projektih, lahko vodja tima zavzame tudi vlogo produktnega vodje. Pri večjih projektih pa je produktni vodja običajno nekdo izven razvojnega tima, ki je pred metodo Scrum uvedbo opravljal funkcijo projektnega managerja. Strle poudarja, da lahko dober vodja tima opravlja vse tri vloge: skrbi za ustrezno izvajanje metode Scrum, usklajuje zahteve naročnika, skrbi za seznam zahtev produkta in vodi razvojni tim. Ločene vloge v praksi vidi kot nesmiselne, saj bi to konkretno pomenilo tri administrativne, nestrokovne vloge na tim 5−8 ljudi. Vloge so bile ločene zgolj na začetku, ko je bila uvedba metode Scrum nova in je bilo še prisotno veliko učenja. Kasneje so v praksi videli, da to ni praktično in so začeli vloge združevati.

Pomembna sprememba po uvedbi je v moči in vplivu razvojnega tima. Tim pridobi na moči in odgovornosti sam upravljati z nalogami znotraj sprinta. Zunanji vpliv ni več legitimen.

4.4.2.4 Spremembe v organizacijski kulturi

Pri prehodu k agilnemu managementu projektov so potrebne spremembe tudi na nivoju organizacijske kulture. Misra, Kumar in Kumar (2010) spremembe vidijo na več nivojih: spremembe stališč povezane z dajanjem prednosti skupinskemu delu, razvoj kulture svobode upravljanja razvoja s strani tima in vključitev naročnika v razvoj. Poleg tega agilna organizacijska kultura spodbuja odprto komunikacijo, zaupanje in deljenje znanja.

Posebnega načrta za upravljanje sprememb povezanih z organizacijsko kulturo ni bilo. Možen razlog vidim v tem, da so bile delno zgornje spremembe v podjetju prisotne že pred uvedbo metode Scrum, kar je olajšalo tranzicijo na novo metodo:

- Principi usmerjenosti k skupinskemu delu, kolaborativnost, delo v manjših timih ter delna odgovornost za projekt v timu so bili prisotni že pred uvedbo nove metode. Vse to pa se je z omejitvijo zunanjega vpliva med sprintom podkrepilo.

- Principi managementa znanja, torej načina posredovanja znanja in učenja, se z uvedbo niso spremenili. Kot prej so bili timi organizirani tako, da se je znanje med zaposlenimi delilo, tako med strokovnjaki kot znotraj večfunkcijskih timov.
- Prav tako niso bile potrebne globje spremembe pri stilu managementa. Stroge hierarhične strukture in direktivni odnosi do razvojnega tima že prej niso bili prisotni. Prav tako je bil management kot pobudnik in podpornik uvedbe pripravljen sprejeti spremembe potrebne za večjo učinkovitost timov.
- Sprememb v stališčih in vrednotah članov tima po uvedbi metode Scrum, npr. povečanje pripadnosti podjetju ali zaupanju v timu, Strle ne zaznava.

4.4.3 Ovire pri uvajanju metode Scrum

4.4.3.1 Vpletanje v delo razvojnega tima

Pred uvedbo metode Scrum je bilo preusmerjanje fokusa razvojnega tima običajna praksa: projektni vodje ali predstavniki prodaje so lahko kadarkoli pristopili do tima s prošnjo ali zahtevo po spremembi trenutnih prioritet dela. Večinoma uspešno. Eden od osnovnih principov metode Scrum je nemoteno in osredotočeno delo tima na vnaprej definiranih uporabniških zgodbah za sprint. Tovrstno vpletanje je torej v nasprotju s tem principom. Vendar pa, razloži Strle, je bilo za pričakovati, da se ta vzorec obnašanja ne bo kar spremenil čez noč. Zato je v uvajalnem času 6 mesecev prevzel vlogo »čuvaja« in blokiral tovrstne interne zahteve, tudi če je ta prišla iz strani managementa. S tem je ščitil tim in poskrbel za to, da se je lahko metoda Scrum kot način dela ukoreninila.

4.4.3.2 Premik k dinamiki skupinskega dela

Uporabniške zgodbe niso definirane zgolj iz tehničnega vidika, ampak večinoma njihova rešitev zahteva kolaboracijo tima. Strle razloži, da so pred uvedbo nekateri strokovnjaki delali izolirano in zgolj oddajali svoj kos naprej naslednjemu v obdelavo. Nekateri so bili pri tem zelo uspešni in učinkoviti. Vendar tega scenarija po metodi Scrum ni več in od vseh članov se pričakuje več komunikacije in timskega dela, več interakcij tudi izven strokovnega področja. Od vsakega posameznika v timu se zahteva prisotnost na dogodkih in izvajanje aktivnosti, kot so planiranje, pisanje uporabniških zgodb in predstavitve dela. Vse to je iz vidika nekega strokovnjaka s specializiranim znanjem lahko smatrano kot dodaten balast, ki ga odvrača od njegovega dejanskega dela, kjer je najbolj učinkovit. Strle je zato pri ožje specializiranih strokovnjakih odpor in nezadovoljstvo že pričakoval.

Omeniti moramo, da se Strle s pozicijo strokovnjakov delno strinja in pravi, da razume, da je za njih bolj učinkovito, če svoj čas posvetijo samo delu, za katerega so edinstveno usposobljeni. Vsi člani tima po metodi Scrum tekom razvoja porabijo več časa za medsebojno usklajevanje, komunikacijo, planiranje, deljenje znanja in podobno. Strle

ocenjuje da kumulativno posamezniki porabijo 10 % delovnega časa znotraj sprinta za aktivnosti, katerih namen je zgolj usklajevanje in ne opravljanja dela. Ta vidik Strle vidi kot negativno plat metode Scrum.

4.4.3.3 Razvojni cikel vključno s testiranjem ni možen v enem sprintu

Predpostavka, da je mogoče v enem sprintu (14 dni) izvesti cel razvojni cikel uporabniške zgodbe vključno s testom, ki je ključni del definicije dokončano, se je izkazala za napačno. Ta ovira v nasprotju z zgornjimi ni bila pričakovana. Razvoj uporabniške zgodbe in njeno testiranje ne moreta potekati istočasno. To ni pomenilo samo časovnih zamikov, ampak tudi slabo izrabo človeških virov.

To so rešili tako, da so testerji in razvijalci na uporabniški zgodbi delali časovno zamaknjeno v ločenih sprintih. V 1. sprintu so razvijalci razvili uporabniške zgodbe v seznamu zahtev sprinta, testerji pa so jih v naslednjem sprintu testirali in definirali morebitne, iz testa izhajajoče naloge ali popravke. To pomeni, da razvojni cikel uporabniške zgodbe ni 1 sprint (14 dni), ampak najmanj 2 sprinta (28 dni). Najhitrejša možna izdaja je torej po 1 mesecu. Nadalje Strle poudarja, da morajo posledično za napake, ki jih testerji odkrijejo v 2. sprintu, najti čas v naslednjem, torej že 3. sprintu. Strle razlog vidi v obsežnejšem testiranju. Pri Microsoft Dynamics AX integracijah govorimo o gotovi programski opremi, kjer se mora testirati vpetost nove uporabniške zgodbe v celoten sistem in ne samo funkcionalnost manjšega dela, ki je bil nadgrajen.

4.5 Management projektov v oddelku

Podjetje se je pri managementu projektov odločilo za kombiniranje metode Scrum z drugimi metodami in orodji. Mohta, Kasat in Yadav (2017) poudarjajo pomen izbora prave metode za implementacijo Dynamics 365 rešitev in navajajo naslednje 3 možnosti:

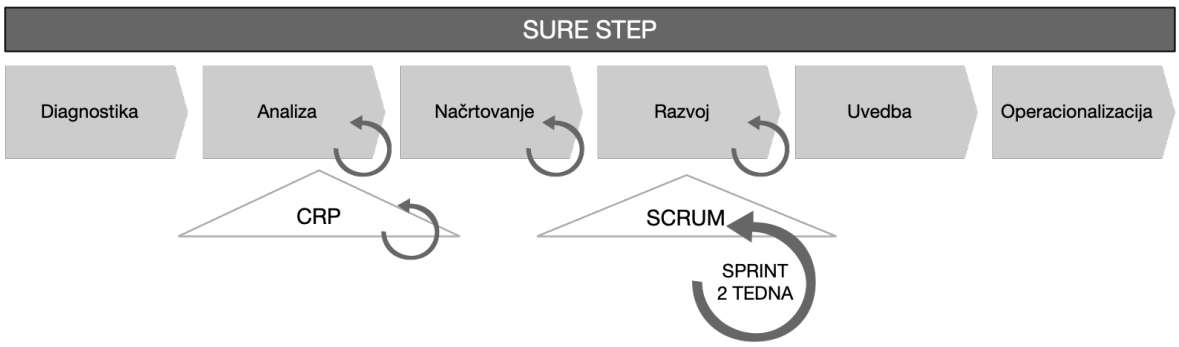
- metoda predstavitve pilotne rešitve (angl. Conference Room Pilot, v nadaljevanju CRP),
- agilna metoda,
- slapovna metoda.

Oddelek uporablja kombinacijo vseh treh. Strle poudarja, da zaradi narave produkta, torej prilagoditve gotove rešitve v nasprotju z razvojem nove programske rešitve, in narave posla, kjer je podjetje prodajalec programske opreme in tako močneje pogodbeno zavezan omejitvam železnega trikotnika, metoda Scrum ni primerna za vse faze življenjskega cikla razvoja produkta.

Življenjski cikel razvoja je osnovan na metodologiji MS Sure Step, ki jo je Microsoft razvil prav za projekte implementacije MS Dynamics rešitev in poteka skozi naslednje faze: diagnostika, analiza, načrtovanje, razvoj, uvedba (angl. deployment) in operacionalizacija. Sure Step temelji na slapovni metodi, kjer si faze sledijo zaporedno in se ne vračajo na

predhodne korake. Vendar pa oddelek uporablja agilno variacijo metode Sure Step, po kateri notranje faze analize, načrtovanja in razvoja potekajo iterativno. Slika 14 prikazuje okviren proces razvoja v oddelku.

Slika 14: Oris procesa implementacije informacijske rešitve v oddelku



Prirejeno po Aamer (2011).

4.5.1 Analiza poslovnih potreb in definiranje zahtev

V začetnih fazah (diagnostika in analiza) metoda Scrum ni v uporabi. Delo v začetnih fazah projekta je bolj podobno planski kot pa agilni metodi, saj vključuje poglobljeno analizo in podrobno načrtovanje.

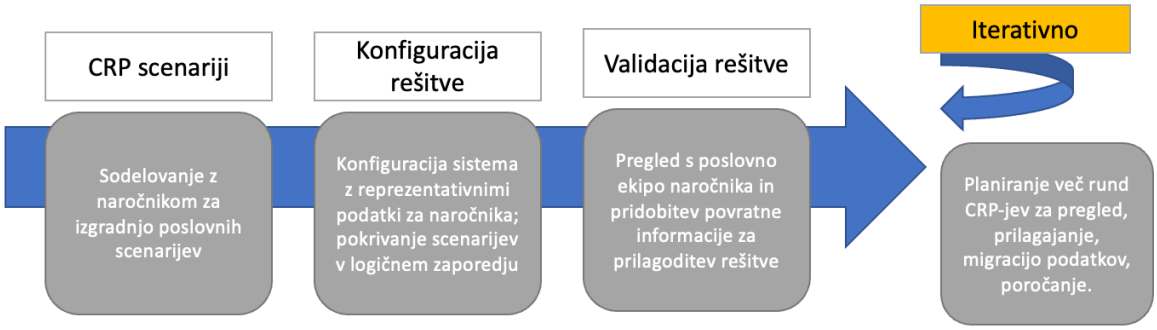
Strle v teh fazah agilne metode smatra kot neprimerne zaradi naslednjih razlogov:

- Informacijska rešitev se razvija za naročnika v nasprotju z internim razvojem, zato je pomembno še pred začetkom implementacije natančno oceniti zahtevnost, stroške in potreben čas. V nasprotju z agilnimi metodami, kjer se zahteve natančneje definirane šele med razvojem in se cilji definirajo preko sprotne komunikacije z naročnikom, pa se tukaj v praksi podjetje na podlagi ocen tem okvirom pogodbeno zaveže. Spremembe tako pomenijo tudi spremembo pogodbe.
- Osnovna informacijska rešitev je že razvita in se zgolj prilagaja naročniku. Implementacija vključuje nastavitve in dodelave specifične za poslovne potrebe. Te dodelave se nanašajo na konkretne procese podjetja naročnika. V nasprotju z razvojem novega produkta ali programske opreme, kjer med razvojem končni cilj/končni produkt ni nujno jasno viden, se tukaj pričakuje manj odstopanj od zadanih ciljev in zahtev med razvojem. Zato je smiselno na začetku vložiti več časa in truda v analizo in načrtovanje končnega produkta še pred začetkom implementacije.

Po diagnostiki procesov in infrastrukture naročnika ter njihovih poslovnih zahtev za definiranje obsega potrebnih funkcionalnosti uporabljajo metodo CRP. Potek je prikazan na sliki 15. Metoda predvideva pripravo prototipa programske opreme, torej funkcionalnosti delujočega sistema, ki se ga predstavi ključnim osebam in odločevalcev na strani naročnika že v fazi načrtovanja, pred implementacijo. Cilj je, da se zgodaj v razvoju odkrije morebitne

napačno identificirane potrebe, napačne predpostavke, napake v načrtu, nepotrebne funkcionalnosti in podobno. Naročnik nima vedno jasne slike želene rešitve, zato ji vpogled v delovanje ERP-rešitve omogoča lažjo predstavo funkcionalnosti in pogodbeno potrditev le-teh. Na drugi strani pa se mora ponudnik že zgodaj v procesu poglobiti v procese naročnika in ima s pilotom možnost to naročniku tudi pokazati.

Slika 15: Koraki CRP-metode



Prirejeno po Mohta, Kasat & Yadav (2017).

Na skupni delavnici ponudnik na primeru prototipa rešitve predstavi želene funkcionalnosti. Predstavijo osnovne funkcionalnosti Microsoft Dynamics ERP-rešitve z vzorcem naročnikovih podatkov in njim prilagojeno konfiguracijo. Morebitne dodelave sicer še niso razvite, ampak si naročnik z vizualno podprtim opisom funkcionalnosti te lažje predstavlja. Rezultat so zahteve, katerim se zaveže tako naročnik kot izvajalec v pisnem dokumentu. Ta dokument je nato osnova za pripravo uporabniških zgodb v seznamu zahtev produkta. Proces je iterativen.

Koristi CRP-metode vključujejo:

- Potrditev pravilnega razumevanja poslovnih potreb in zahtev pred začetkom razvoja.
- Jasno in zgodaj definiran obseg projekta in zahteve, kar pomeni manjše možnosti širitve obsega, saj so bili tako procesi, ki jih bo podpiral sistem, kot tudi cilji in pričakovanja glede funkcionalnosti pisno definirani. Dokument nato služi kot vodilo k cilju in omogoča zavračanje morebitnih kasnejših zahtev po (neutemeljenih) dodatnih funkcionalnostih. To ščiti tako ponudnika kot odločevalce na strani naročnika.
- Razvojni tim zgodaj v procesu bolje spozna poslovne procese naročnika.
- Naročnik se prej spozna s sistemom in lahko prej vidi koristi, ki jih le-ta prinaša.
- Morebitne potrebe po spremembah v načrtu informacijske rešitve se odkrijejo zgodaj in pravočasno, torej še pred implementacijo, ko bi spremembe imele bistveno večji negativen vpliv na stroške in čas razvoja.

4.5.2 Iterativni razvoj po metodi Scrum

Razvoj poteka po metodi Scrum in prinaša med drugim koristi boljše interne komunikacije, zmanjšanega tveganja z zaznavanjem sprememb zgodaj, predvsem pa prinaša organizacijske koristi. Strle poudarja, da z metodo Scrum držijo tempo dela. Delajo na manjših, bolj obvladljivih kosih, predvsem pa so timi med seboj usklajeni in časovno poravnani. Organizacijske koristi vidi kot največji doprinos uvedbe te metode.

Po zajetju zahtev po metodi CRP pričnejo z razvojem. Po približno 2 mesecih razvijejo osnovno konfiguracijo programske rešitve, zapolnijo jo s podatki naročnika, predstavijo in zajamejo povratne informacije. Razvoj poteka inkrementalno po metodi Scrum, vendar je prva izdaja naročniku šele po 2 mesecih. Strle razloži, da je šele takrat mogoče pokazati delujoč del programske opreme. Hitrejša izdaja se je izkazala za nemogočo pri obsegu implementacije MS Dynamics AX rešitve. Zato so se odločili na tem mestu odstopati od metode Scrum, ki predvideva inkrement primeren za izdajo po vsakem sprintu.

To osnovno različico pregledajo skupaj z naročnikom in pridobijo povratne informacije. V nasprotju z metodo Scrum, kjer se zahteve oblikujejo tekom razvoja, naročnik že tukaj pisno potrdi funkcionalnosti prikazane ali nakazane v osnovnem modelu. V primeru odklonov od prvotnih zahtev se prične pogajanje za zamenjavo zahtev na način, da obseg projekta ostaja enak.

Iz zajetih zahtev se spišejo uporabniške zgodbe in shranijo v seznamu zahtev produkta. Seznam zahtev vključuje tako velike, epske zgodbe, grobo definirane zgodbe kot že razdelane uporabniške zgodbe, primerne za vključitev v sprint. Če zahteva ni v seznamu zahtev produkta, ta ne more biti vključena v sprint. Pogoj za vključitev v sprint pa je tudi doseganje kriterijev pripravljenosti. Ta definira naslednje pogoje:

- opis funkcije je v kratki in jedrnati obliki,
- vključuje skico nove maske, obrazca in podobno,
- vključuje kriterije sprejetja (angl. acceptance criteria),
- je izvedljiva v enem sprintu oz. dveh vključno s testiranjem.

Enkrat tedensko se razvojni tim in produktni vodja sestanejo na sestanku razčlenitve zgodb, katerega cilj je uporabniške zgodbe razdelati in pripraviti na morebitno vključitev v sprint. To pomeni razbiti večje uporabniške zgodbe na manjše, identificirati manjkajoče informacije in uporabniških zgodbam določiti težavnost izvedbe. Ocena težavnosti ni v časovnih enotah, ampak v točkah. Pri določanju točk preko »pokra planiranja« sodelujejo vsi člani razvojnega tima.

Pred sprintom se razvojni tim, skrbnik metode in produktni vodja sestanejo za planiranje sprinta, katerega rezultat je seznam zahtev sprinta. Produktni vodja predstavi za sprint primerne uporabniške zgodbe v seznamu zahtev produkta in njihove prioritete. Razvojni tim določi kombinacijo zgodb, ki lahko v enem oz. dveh sprintih s testiranjem, tvorijo

funkcionalen inkrement produkta. Razvojni tim se vsebini seznama zahtev sprinta zaveže, vendar Strle poroča, da pri izvedbi niso vedno uspešni in ne implementirajo vseh zadanih uporabniških zgodb v zadanem roku. Razlog vidi v tem, da se pri razvoju uporabniške zgodbe pogosto med sprintom izkaže, da je težavnost višja od predvidene ali da so potrebne dodatne informacije iz strani naročnika, na katere se mora nato počakati. Skladno z metodo Scrum se tim zjutraj udeležuje dnevnih sestankov, ki so kratki, dolgi največ 15 minut. Vsak član na hitro poroča, na kaj je delal včeraj, na katerih uporabniških zgodbah bo delal danes in kje ima morebiti težave.

Sprint ima standardno dolžino 14 dni. V nasprotju s smernicami po metodi Scrum pa na koncu vsakega sprinta ni na voljo novo razvita funkcionalnost, ampak je cikel izdaj zaradi zamika testiranja uporabniške zgodbe v sledeči sprint najmanj 28 dni. Uporabniške zgodbe so smatrane kot končane šele, ko dosežejo definicijo dokončanega**.** Definicija dokončanega vključuje naslednje pogoje:

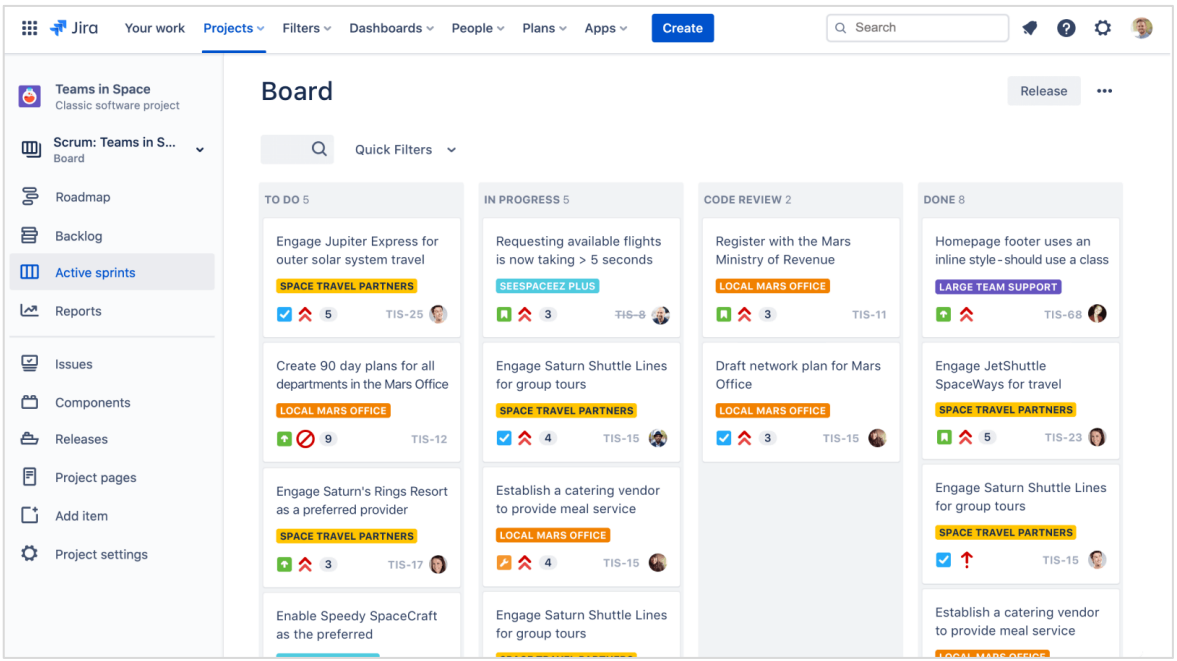
- ni opozoril ali napak pri preverbi kode,
- koda je opremljena s komentarji in skladna s smernicami kodiranja,
- interna preverba kode (drug razvijalec),
- uspešno prestano avtomatizirano preverjanje funkcionalnosti,
- kriteriji sprejetja so uspešno opravljeni,
- funkcionalnost iz vidika uporabnika je preverjena in varnostna dovoljenja (angl. security permissions) so ustrezna,
- funkcionalnost v skupnem okolju je preverjena,
- funkcionalnost je bila predstavljena svetovalcu.

Ob koncu sprinta razvojni tim na pregledu sprinta predstavi delo produktnemu vodji in drugim deležnikom. Namen je potrditev (ali zavrnitev) narejenega in pa pridobivanje povratnih informacij za nadaljnji potek razvoja. Sestanek traja do 1 ure. Sprint pregledu sledi sprint retrospektiva, ki jo Strle označi kot enega izmed aktivnosti po metodi Scrum z največ dodane vrednosti. Skupaj s skrbnikom metode razvojni tim kritično pregleda svoje delo v sprintu in predlaga optimizacije. Pogovorijo se tudi o ovirah, na katere so naleteli tekom sprinta in možnostih odprave le-teh v prihodnje. Tipični primeri takšnih izboljšav so bili planiranje več časa za morebitna dodatna dela, razbitje uporabniških zgodb na še manjše enote, boljša opredelitev uporabniških zgodb pred vključitvijo v sprint.

Za upravljanje seznama zahtev produkta in sprinta uporabljajo programsko opremo Jira, ki omogoča shranjevanje in urejanje uporabniških zgodb v tabelarni obliki z opredelitvijo prioritet v obliki barvnih puščic in vnosa ocene obsega dela. S klikom na uporabniško zgodbo je možno videti in urejati njene podrobnosti. Zgornji sklop predstavlja aktualni seznam zahtev sprinta, spodaj pa se nahaja seznam zahtev produkta. Uporabniške zgodbe je možno razčlenjevati na manjše dele kot tudi združevati v verzije.

Možen pa je tudi pogled v obliki table po metodi Scrum prikazan na sliki 16. Digitalna oblika table kot fizična omogoča pregled in premike kartic ter mnogo drugih prednosti digitalne oblike. Strle omeni, da so pri prvi uvedbi metode Scrum v podjetju poskusili delati s fizično tablo in samolepilnimi listki, vendar jim to glede na tehnologije, ki so na voljo, ni prineslo dodane vrednosti.

Slika 16: Primer table po metodi Scrum v programu Jira



Vir: Atlassian (brez datuma a).

Vse aktivnosti po prehodu v živo (trening uporabnikov, podpora) se ne izvajajo po metodi Scrum. Zaradi potrebe po večji odzivnosti, torej prej kot v roku najkrajšega možnega sprinta (1 teden), so uvedbo metode Scrum v to zaključno fazo opustili.

4.6 Ključne ugotovitve

4.6.1 Uspešnost projektov po uvedbi metode Scrum

V prvem sklopu se sprašujem ali so v praksi projekti razvoja programske opreme uspešnejši po uvedbi metode Scrum. Uspešnost projektov ocenjujem po štirih kriterijih osnovanih na Gollner in Baumane-Vitolina (2016) modelu uspešnosti projektov: projektni management, čas in proračun, kvaliteta ERP-rešitve, zadovoljstvo uporabnikov.

4.6.1.1 Projektni management

Strle največji pozitivni doprinos metode Scrum vidi v organizacijskih koristih in vplivu le-teh na projektni management. Pregled nad projektom je izboljšan. Skozi definirane časovne sklope in s tem časovne usklajenosti timov je možno bolj učinkovito planiranje in morebitna menjava zahtev med timi. Uporabniške zgodbe so manjši in bolj obvladljivi kosi dela, na katere se ekipa lažje osredotoča. S tem je omogočen trajnosten tempo dela, ki skrbi za konstanten napredek.

K preglednosti in usklajenosti udeležencev veliko doprinesejo dnevni sestanki, s katerimi zagotavljajo, da so vsi na tekočem in da se morebitni problemi hitreje identificirajo in rešijo. Veliko vlogo Strle daje retrospektivi. Na tem sestanku se ciljno razrešijo tudi težave organizacijske narave, ki jih sicer brez tega sestanka ne bi aktivno reševali.

Del projektnega managementa je tudi komunikacija, ne samo znotraj tima, ampak z vsemi deležniki. Naročnik je zdaj bolj udeležen v razvoj in komunikacija preko produktne vodje poteka konstantno. Tudi drugi deležniki so zaradi večje preglednosti nad projektom bolj informirani o stanju in napredku le-tega.

4.6.1.2 Čas in proračun

Čas in proračun sta med seboj povezana pojma, podaljšanje projekta pomeni tudi povečanje stroškov. Pred uvedbo metode Scrum so bile v poznih fazah projekta pogosto najdene napake, ki so projektu dodale mesece razvoja in posledično podjetju povzročile višje stroške, medtem ko je cena naročniku ostala nespremenjena. Te napake so zdaj najdene prej ali pa niti ne nastanejo.

4.6.1.3 Kvaliteta produkta

Tudi pred uvedbo metode Scrum je implementirana rešitev dosegala enako stopnjo kvalitete kot po uvedbi. Razlika je zgolj v tem, da je zdaj zaradi izboljšane preglednosti nad projektom in izboljšanimi strukturami dela lažje priti do tega rezultata.

H kvaliteti implementirane rešitve doprinesejo natančneje definirani kriteriji dokončanega pravi Strle. S takšnim jasno definiranim kontrolnim seznamom in sprotnim testiranjem so negativna presenečenja na koncu manj verjetna.

4.6.1.4 Zadovoljstvo uporabnikov

Strle potrdi, da so naročniki zdaj bistveno bolj zadovoljni, saj dobijo programsko rešitev predstavljeno zgodaj v razvoju, jo spremljajo tekom razvoja in pri prehodu v živo je programska rešitev praktično končana in ustreza njihovim pričakovanjem.

Trije kriteriji uspešnosti so potrjeni (projektni management, čas in proračun, zadovoljstvo uporabnikov). Četrti kriterij, ki se nanaša na izboljšanje kvalitete programske opreme ni potrjen, vendar tudi pri tem ne gre za zanikanje dobre kvalitete rešitve, zgolj, da se je podjetje tudi pred uvedbo metode Scrum prizadevalo dosegati strog nivo kvalitete, tudi če je to pomenilo več sredstev. Iz tega sklepam, da so v primeru preučevanega oddelka projekti uspešnejši po uvedbi metode Scrum.

4.6.2 Prilagajanje metode potrebam projekta

V drugem sklopu se sprašujem ali je za učinkovito uporabo metode Scrum potrebno metodo prilagoditi konkretnim potrebam podjetja v praksi. Tudi to lahko potrdim. Oddelek metode Scrum ni uvedel celostno, ampak je metodo prilagodil potrebam implementacije rešitve Microsoft Dynamics AX:

- Uporabniške zgodbe se zaradi pogodbenih omejitev manj prilagajajo tekom razvoja kot bi bilo to predvideno po metodi Scrum. Naročnik se ob začetku implementacije pisno zaveže k definiranemu obsegu.
- Uporabniška zgodba ni razvita in testirana v 1 ampak v 2 sprintih.
- Uporabniške zgodbe so lahko napisane tudi iz vidika funkcije brez vključitve uporabniške perspektive.
- Vloga skrbnika metode in vodje tima je združena v eni osebi.

Poleg zgoraj opisanih prilagoditev pa oddelek ne uporablja metode Scrum v celotnem življenjskem ciklu razvoja, ampak samo v koraku implementacije. Zaradi potrebe po natančnih stroškovnih in časovnih ocenah, je na začetku procesa potrebno obsežnejše analiziranje in načrtovanje. Na podlagi rezultatov ocen se pripravi zavezujoča se ponudba. Kljub tem odstopanjem od metode Scrum v čisti obliki, management projektov poteka tekoče in implementacije programske rešitve pri naročnikih so uspešne.

4.6.3 Omejitve

Omejitve magistrske naloge in priložnosti za nadaljnje raziskave vidim v njenem vzorcu. Pridobljeni podatki iz enega podjetja nam dajejo vpogled in boljše razumevanje metode Scrum v praksi, vendar pa rezultatov ne moremo posploševati. Priložnost za nadaljnje raziskovanje vidim v razširitvi vzorca, v raziskovanju uspešnosti projektov po Scrumu in uvedbe metode Scrum v nadaljnjih podjetjih.

Poglobljeni intervju ima mnoge prednosti, med njimi pridobivanje poglobljenih informacij za boljše razumevanje tematike preko možnosti postavljanja dodatnih vprašanj ter visoke relevantnosti informacij zaradi skrbnega izbora intervjuvanca. Med omejitve te raziskovalne tehnike pa spada subjektivnost odgovorov, intervjuvanec odgovarja iz svojega edinstvenega vidika. Intevjuvala sem osebo na vodstveni poziciji kar ima morebiti vpliv na izbor in

interpretacijo podanih informacij. Zanimivo bi bilo videti kako so uvedbo metode Scrum doživljali člani timov v oddelku. Nadaljnja raziskava stališč do metode Scrum med člani tima bi dala bolj celostno sliko na uporabo in uvedbo metode Scrum v oddelku.

Prav tako bi dodaten vpogled v uspešnost projektov po uvedbi metode Scrum doprinesel vidik naročnika. Le-ta bi lahko podal oceno ekonomske vrednosti, ki jo je prinesla implementacija informacijske rešitve.

SKLEP

Podjetja se pri managementu projektov srečujejo z mnogimi izzivi, predvsem pa je za uspešnost potrebno konstantno prilagajanje, potrebna je agilnost. Planske metode sicer prinašajo višjo predvidljivost ali zgolj iluzijo le-te; vendar pa projekti vodeni po teh metodah beležijo vse slabše odnose z naročnikom, težje upravljajo s spreminjajočimi vplivi okolja in zahtev ter pričakovanim vedno hitrejšim tempom razvoja. Mnoga IT-podjetja rešitev vidijo v metodi Scrum, ki obljublja prilagodljivost. Z iterativnim načinom dela in konstantno komunikacijo z naročnikom ali uporabniki, metoda Scrum med drugim obljublja večjo preglednost in učinkovitost projektnega managementa ter izboljšanje kvalitete informacijske rešitve z upoštevanjem dejanskih potreb uporabnikov. Pomembno vlogo ima osebna interakcija. Scrum in druge agilne metode dajejo prednost direktni in pogosti komunikaciji z deležniki, predvsem z naročnikom; zlasti pa se ne zanašajo na dokumentacijo. Rezultat so informacijske rešitve, ki bolje zadovoljijo naročnikova pričakovanja.

Učenje iz predhodnih iteracij in informacij iz okolja je pomemben del metode Scrum. Kot omenita Quigley in Pries (2011), metoda vključuje aktivnosti, ki temeljijo na učenju, in omogoča, da se naučeno uporabi v nadaljnjem razvoju. Vsak cikel projekta pridobiva znanje skozi sprint pregled in retrospektivo ter povratno informacijo od naročnika.

Uvedba metode Scrum v preučevanem oddelku podjetja Adacta je bila uspešna in je skladno z ugotovitvami avtorjev Ceschi, Sillitti, Succi in De Panfilis (2005) prinesla tako izboljšanje managementa samega procesa razvoja kot izboljšanje odnosov z naročniki. Pod definiranimi kriteriji uspešnosti projektov so bile spremembe pozitivne.

Projektni management je zaradi inkrementalnega načina dela, definiranih ciklov, med seboj usklajenih timov in boljšega pregleda zahtev bolj obvladljiv in pregleden. Viri oddelka kot so stroški, čas in človeški viri so bolje upravljani, saj so izzivi odkriti in rešeni v zgodnji fazi razvoja in ne kot prej šele ob koncu razvoja. Projekti ob zaključku tako zahtevajo bistveno manj dodatnih neplaniranih sredstev. Veliko k temu rezultatu prispeva iterativna in inkrementalna narava metode Scrum. Delo se v definiranih časovnih intervalih ne samo razvija, ampak tudi testira in usklajuje z naročnikom.

Komunikacija med naročnikom in razvojnim timom je ključna za uspeh projekta. Naročnik je z uvedbo pogoste povratne zanke informacij bolj vključen v proces razvoja in podjetje

tako lažje poskrbi za to, da so njegova pričakovanja izpolnjena. Več informacij in izpolnitev pričakovanj pa vodi v višje zadovoljstvo naročnika. Edini kriterij uspešnosti, ki ni bil potrjen, je bilo izboljšanje kvalitete informacijske rešitve, kjer je podjetje že prej poskrbelo za to, da je implementirana rešitev dosegala zadane standarde kvalitete, tudi če je to pomenilo več prvotno nepredvidenega dela.

Pri uvedbi metode Scrum je omembe vredno, da so bili pogoji za uvedbo v oddelku ugodni. Metoda je bila pred tem že uvedena v drugem oddelku podjetja, iz katerega so lahko črpali informacije in izkušnje. Organizacijska kultura je bila vsaj delno že usklajena z vrednotami metode Scrum, uvedba pa je imela polno podporo vodstva.

Oddelek metode Scrum ni uvedel v celoti, ampak je njegov pristop bližje inženiringu metode, ki ga Fitzgerald, Harnett in Conboy (2006) opredelijo kot izbor delov različnih metod, ki so najbolj primerne za posamezne procese razvoja. Iz njih nastane nova prilagojena metoda. Cobb (2015) poudarja, da ustrezni rezultati niso mogoči pri izboru ene metode za vse vrste projektov. Plansko usmerjene in agilne metode niso ali dobre ali slabe, učinkovite ali neučinkovite, ampak je njihova ustreznost odvisna od potreb projekta. Prav tako se usmerjenost k planu in agilnosti ne izključujeta.

V tem smislu je bila tudi v oddelku metoda Scrum implementirana zgolj v fazi razvoja, v fazah analize in načrtovanja pa je zaradi potreb priprave ponudbe za naročnika z definirano ceno, obsegom in roki ohranjeno obsežno analiziranje in planiranje, značilno za planske metode. Metoda Scrum v polni obliki namreč ni nujno primerna za projekte s strogimi časovnimi in stroškovnimi okviri. Iz tega lahko sklepamo, da metoda Scrum prinaša pomembne pozitivne spremembe in doprinaša k poslovnemu uspehu, tudi če ni implementirana v celoti. Za uspešno uvedbo metode je bilo v predstavljenem primeru dele metode potrebno prilagoditi konkretnim potrebam projektov.

LITERATURA IN VIRI

1. Aamer, M. (2011, 23. november). *Microsoft Dynamics Sure Step Agile Implementation.* Pridobljeno 30. septembra 2021 iz https://social.technet.microsoft.com/wiki/contents/articles/5768.microsoft-dynamics-sure-step-agile-implementation.aspx
2. Ahmad, M. O., Markkula, J. & Oivo, M. (2013). Kanban in software development: A systematic literature review. Pridobljeno 10. septembra 2021 iz https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6619482
3. Atlassian. (brez datuma a). *Jira Software*. Pridobljeno 1. septembra 2021 iz https://www.atlassian.com/software/jira
4. Atlassian. (brez datuma b). *Working with Sprints*. Pridobljeno 2. septembra 2021 iz https://confluence.atlassian.com/display/GH062/Working+with+Sprints

5. 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., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001). *Manifesto for Agile Software Development*. Pridobljeno 13. septembra 2020 iz https://agilemanifesto.org
6. Be-terna. (brez datuma). *Dynamics 365 for finance and supply chain*. Pridobljeno 22. avgusta 2021 iz https://www.be-terna.com/sl/resitve/erp/finance-and-supply-chain
7. Bulao, Jacquelyn (2021). *How fast is technology advancing in 2021* [objava na blogu]. Pridobljeno 7. oktobra 2021 iz https://techjury.net/blog/how-fast-is-technology-growing/#gref
8. Cao, L., Mohan, K., Xu, P. & Ramesh, B. (2009). A framework for adapting agile development methodologies. *European Journal of Information Systems* (18), 332–343.
9. Ceschi , M., Sillitti, A., Succi, G. & De Panfilis, S. (2005). Project Management in Plan-Based and Agile Companies. *IEEE Software, 3*(22), 21–27.
10. Chemuturi, M. (2013). *Mastering IT Project Management : Best Practices, Tools and Techniques.* Plantation: J. Ross Publishing.
11. Cobb, C. G. (2015). *The Project manager's guide to mastering agile. Principles and Practices for an Adaptive Approach.* Hoboken: John Wiley & Sons, Incorporated.
12. Cohn, M. (2010). *Succeeding with Agile. Software development using Scrum.* Upper Saddle River: Addison-Wesley.
13. DeLone, W. H. & McLean, E. R. (2003). The DeLone and McLean Model of Information Systems Success: A Ten-Year Update. *Journal of Management Information Systems, 19*(4), 9–30.
14. Denning, S. (2011). Successfully implementing radical management at Salesforce.com. *Strategy & Leadership, 39*(6), 4–10.
15. Fitzgerald, B., Hartnett, G. & Conboy, K. (2006). Customising agile methods to software practices at Intel Shannon. *European Journal of Information Systems, 15*(2), 200–213.
16. Fowler, M. & Highsmith, J. (2001). The agile manifesto. *Software Development, 9*(8), 28–35.
17. Freedman, R. (2009). *The roots of agile project management* [objava na blogu]. Pridobljeno 12. februarja 2021 iz https://www.techrepublic.com/blog/tech-decision-maker/the-roots-of-agile-project-management/
18. Gollner, J. A. & Baumane-Vitolina, I. (2016). Measurement of ERP-Project Success: Findings from Germany and Austria. *Engineering Economics, 27*(5), 498–508.
19. Holcombe, M. (2008). *Running an Agile Software Development Project.* Hoboken: Wiley.
20. Ionel, N. (2008). Critical Analysis of the Scrum Project Management Methodology. *Annals of the University of Oradea, Economic Science Series, 17*(4), 435–441.
21. Kniberg, H. (2015, 13.maj). *Scrum and XP from the Trenches. How we do Scrum.* Pridobljeno 6. februarja 2021 iz https://www.infoq.com/minibooks/scrum-xp-from-the-trenches-2/.

22. Koch, A. S. (2005). *Agile Software Development: Evaluating the Methods for Your Organization.* Boston: Artech House, Inc.
23. Mahnič, V. & Urevc, J. (2012). Ocena prednosti metode Scrum in njenih tipičnih praks. *Uporabna Informatika, 20*(3), 184–194.
24. Measley, P. (2015). *Agile Foundations: Principles, Practices and Frameworks.* London: The Chartered Institute for IT.
25. Misra, S. C., Kumar, V. & Kumar, U. (2010). Identifying some critical changes required in adopting agile practices in traditional software development projects. *International Journal of Quality & Reliability Management, 27*(4), 451–474.
26. Mohta, R., Kasat, Y. & Yadav, J. (2017). *Implementing Microsoft Dynamics 365 for Finance and Operations.* Birmingham: Packt Publishing.
27. Ozierańska, A., Skomra, A. & Dorota, K. (2016). The Critical Factors of Scrum Implementation in IT Project - the Case Study. *Journal of Economics and Management, 25*, 79–96.
28. Petter, S., DeLone, W. & McLean, E. R. (2013). Information Systems Success: The Quest for the Independent Variables. *Journal of Management Information Systems, 29*(4), 7–62.
29. Project Management Institute. (2004). *A guide to the Project Management Body of Knowledge* (3.izdaja). Newtown Square: Project Management Institute, Inc.
30. Quigley, J. M. & Pries, K. H. (2011). *Scrum Project Management.* Boca Raton: CRC Press.
31. Rubin, K. S. & Lichtenberg, K. (2014). *Essential Scrum: Umfassendes Scrum-Wissen aus der Praxis.* Heidelberg: MITP.
32. Scheucher, G. A. (2017). *Strategies to Promote IT Project Success* (doktorska disertacija). Minneapolis: Walden University.
33. Standish Group International, Inc. (2015). *Chaos Report 2015.* Pridobljeno 7. junija 2021 iz https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf
34. Takeuchi, H. & Nonaka, I. (1986). The new new product development game. *Harvard Business Review, 64*(1), 137–146.
35. Verzuh, E. (2011). *The Fast Forward MBA in Project Management.* Hoboken: John Wiley & Sons, Incorporated.
36. Walle, A. H. (2015). *Qualitative Research in Business : A Practical Overview.* Newcastle upon Tyne: Cambridge Scholars Publishing.
37. Wysocki, R. K. (2006). *Effective Software Project Management.* New York: John Wiley & Sons, Incorporated.
38. Wysocki, R. K. (2009). *Effective Project Management: Traditional, Agile, Extreme.* Indianapolis: Wiley Publishing, Incorporated.