UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

ZAKLJUČNA STROKOVNA NALOGA VISOKE POSLOVNE ŠOLE

ANALIZA PRIMERNOSTI METODOLOGIJE RAZVOJA ZA DIGITALNO PLATFORMO

Ljubljana, maj 2020 JURE ŠAVRON

IZJAVA O AVTORSTVU

Podpisani Jure Šavron, študent Ekonomske fakultete Univerze v Ljubljani, avtor predloženega dela z naslovom Analiza primernosti metodologije razvoja za digitalno platformo, pripravljenega v sodelovanju s svetovalcem red. prof. dr. Alešem Groznikom

I Z J A V L J A M

1. da sem predloženo delo pripravil 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 poskrbel, 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 pridobil vsa potrebna dovoljenja za uporabo podatkov in avtorskih del v predloženem delu in jih v njem jasno označil;

7. da sem pri pripravi predloženega dela ravnal v skladu z etičnimi načeli in, kjer je to potrebno, za raziskavo pridobil 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 _____ Podpis študenta: _____

KAZALO

KAZALO TABEL

KAZALO SLIK

UVOD

V svetu informacijske tehnologije se neprestano pojavlja želja po izboljšanju razvoja programske opreme, da bi bil razvoj učinkovitejši, hitrejši in cenejši. In izboljšujejo se metodologije vodenja projektov, orodja, znanja, poznavanje dobrih praks. Pojav agilnih metodologij je v svetu naredil velik preobrat pri razvijanju digitalnih produktov. Iz agilnih metodologij se je razvila tudi metoda SCRUM, ki je bistvo tega diplomskega dela. V diplomski nalogi je opisan način razvijanja programske opreme po tradicionalni (angl. Waterfall) ali v nadaljevanju slapovni model. Vso teorijo sem preveril tudi s polodprtimi intervjuji, ki so mi omogočili podroben in natančen vpogled v razvoj programske opreme.

Namen diplomske naloge je predstaviti tradicionalne in agilne metode, prikazati način delovanja metod in predstaviti njihova osnovna načela. V diplomski nalogi je podrobno opisan tudi razvojni okvir SCRUM.

Cilj naloge je ugotoviti, zakaj so programske rešitve, razvite po agilni metodologiji SCRUM, uporabnejše in učinkovitejše od programske opreme, razvite po tradicionalni metodi modela slapov.

Diplomsko delo je razdeljeno na tri dele. V prvem delu sem se osredotočil na značilnosti samega projekta ter na to, kaj je projektno vodenje, kakšne naloge ima projektni vodja in kaj je potrebno, da lahko nekaj sploh poimenujemo projekt.

V drugem delu sem razložil, kaj je agilnost, kaj nam omogoča in predstavil njene značilnosti. Razložil sem tudi značilnosti tradicionalnih metod in modela slapov.

V tretjem delu sem želel z intervjuji polodprtega tipa ugotoviti, zakaj je agilna metodologija SCRUM uspešnejša od tradicionalne metode modela slapov.

1 TEORETIČNI DEL

1.1 Kaj je projekt

Definicijo projekta lahko razdelimo na tiste, ki projekt opredeljujejo kot časovni in ciljno usmerjen proces, in na tiste, ki poudarjajo vlogo oziroma namero projektov. Med definicijami zasledimo mednarodne ali dogovorjene definicije različnih standardov.

Cleland (1999) razlaga projekt kot kombinacijo potencialov v organizaciji, združenih z namenom ustvariti določeno novost, ki bo podjetju pomagala pri uresničevanju strategije. Vsi projekti imajo določen ciklus in potekajo po zaporedju posameznih faz.

Lewis (1998) razlaga projekt kot delo, ki se izvede samo enkrat. Imeti mora jasen začetek in konec ter opredeljena proračun in načrt. To so nujne zadeve, ki jih je treba postaviti v izhodiščni cilj.

Project Management Institute, Inc. (2004, str. 5) razlaga projekt kot začasno prizadevanje z začetkom in koncem za ustvarjanje unikatnega izdelka, storitve ali rezultata. To pomeni, da je projekt sklop aktivnosti, ki morajo imeti določen cilj. Projekt se običajno definira na samem začetku, nato se s časom ponovno pregleda in definira.

Omenjenih je le nekaj definicij projekta. V literaturi sem jih zasledil kar veliko. Če povzamem vse definicije projekta in to sestavim v neko smiselno obliko, bi projekt definiral kot sodelovanje med zaposlenimi v neki organizaciji ali podjetju s sredstvi in z aktivnostmi, za katerega je značilno, da se ne ponavlja in katerega proizvod ali storitev je unikatna in je časovno omejena. V projektu potemtakem morajo sodelovati sredstva in zaposleni.

1.2 Značilnosti projekta

- **Začasnost in končnost**

Projekt mora biti **začasen in končen**. Končnost si razlagamo kot dejstvo, da se mora vsak projekt po določenem času zaključiti.

Začasnost kot taka se ne na naša na sam produkt, storitev ali rezultat projekta. Veliko projektov je narejenih za ustvarjanje nekega dolgoročnega izida. Projekti imajo velikokrat tudi načrtovane ali nenačrtovane socialne, ekonomske ali okolijske vplive, ki se pojavijo potem, ko se je projekt že davno zaključil (Rozman & Stare, 2008).

- **Enkratnost projekta**

S projektom običajno želimo ustvariti neki izid. To so lahko proizvodi, storitve ali rezultat. Enkraten je tudi, ker je zelo malo verjetno, da bo projekt ponovljen na enak način ali z istimi udeleženci. Projektno delo je praviloma nerutinsko (Rozman & Stare, 2008).

- **Usmerjenost k cilju ali izid**

Vsak projekt more imeti neki **izid,** kar je lahko produkt, storitev ali rezultat, na primer (Rozman & Stare, 2008):

- proizvod ali artefakt, ki je proizveden, je merljiv in je lahko končni produkt ali pa komponenta nekega drugega produkta;
- zmožnost izvajati storitve, kot je poslovna funkcija, ki podpira proizvodnjo ali distribucijo;
- rezultat, ki je izid ali dokument, na primer raziskava, ki je lahko uporabljena za ugotovitev, če je neki trend na trgu prisoten ali če bo neki proces koristil družbi.

Vsak projekt je treba graditi postopno in ga prilagajati po potrebah, ki se pojavijo (Rozman & Stare, 2008).

– **Omejenost**

Projekti so omejeni na različne načine in omejenost med njimi zelo variira. Med glavne omejitve štejemo kakovost, končni rok in finančna sredstva. Na omejenost vplivajo tudi zakonodaja, etika, lokalna skupnost in socialni vidik. Med omejitve štejemo tudi ljudi. Vsak človek namreč nima enakih sposobnosti in znanja (Rozman & Stare, 2008).

– **Kompleksnost**

Vsak projekt ima lahko zelo kompleksne cilje. V veliko primerih v velikih podjetjih projekti zahtevajo sodelovanje med več oddelki ali skupinami ljudi. V projektih se običajno prepleta širok splet ljudi z različnimi znanji, odgovornostmi in s pristojnostmi. Kompleksnost projekta zahteva pazljivo koordiniranje, sledenje rokom, nadzor nad stroški in nadzor nad izvajanjem (Rozman & Stare, 2008).

– **Povezanost in odvisnost projektnih aktivnosti**

Projekt je običajno sestavljen iz niza medsebojno povezanih aktivnosti, ki jih je potrebno izvesti, če želimo doseči želeni cilj. Velikokrat obstaja soodvisnost med različnimi projekti, kar lahko vpliva na dosegljivost kadra v nekem podjetju. Na izvedbo nekega projekta lahko vpliva tudi to, da je za njegovo izvedbo potreben izid nekega drugega projekta (Rozman & Stare, 2008).

– **Konfliktnost**

Projektni menedžerji delujejo v bolj konfliktnem okolju kot drugi menedžerji. Projekt lahko definiramo tudi kot organizacijo v organizaciji. Velikokrat poteka boj za razne vire, kot so zaposleni, finančna sredstva. To je še posebej vidno v organizacijah, kjer se razvija veliko projektov. Nesporazumi se lahko pojavijo zaradi nasprotja interesov naročnika, projektnega tima in javnosti (Rozman & Stare, 2008).

– **Tveganost**

Tveganost je povezana z enkratnostjo in s konfliktnostjo. Vsak projekt je praviloma drugačen in se na prejšnje izide ter procese ne moremo zanašati. Na projekt lahko vplivajo različni dejavniki. Pojavijo se lahko težave z izvedbo, naročniki, vremenom, vplivnimi posamezniki, vodstvom, razmerami na trgu ipd. Vse to lahko vpliva na uspešnost projekta (Rozman & Stare, 2008).

1.3 Projektni menedžment

Pojav projektnega menedžmenta je zelo naraven. Ker so sredstva omejena, stremimo k učinkovitosti, kar pomeni, da s čim manjšim naporom in čim manj sredstvi zadovoljimo čim več svojih potreb. Stremenje k učinkovitosti procesa je pripeljalo do tehnične delitve dela. Ljudje so se v neki organizaciji združevali v združbe, ki so postale samostojne, kvalitativno različne enote. V teh enotah se člani povezujejo, da bi čim bolj smotrno dosegli cilje združbe ter svoje lastne cilje. Kljub vsem prednostim so se pojavile težave. Razlog za težave so seveda razdrobljene naloge, ki so v veliko primerih soodvisne. Zaradi neusklajenosti je prihajalo do neučinkovitosti. Tukaj se je pojavil poklic menedžerja, ki je formalno pripomogel k večjemu pregledu delavnih nalog in formaliziral proces (Rozman & Stare, 2008).

Ravnateljevanje projekta je v metodološkem smislu odločanje, ki sledi splošnemu procesu odločanja. Odločanje se nanaša na povezovanje aktivnosti, določanje rokov, zagotavljanje kakovosti, financiranje projekta in podobno. Pomembo je tudi, da se preprečujejo problemi ter, če se pojavijo, tudi rešujejo. Velikokrat je treba znati poiskati alternativne rešitve in izbrati pravo (Rozman & Stare, 2008).

1.3.1 Vloga projektnega menedžerja

Projektni menedžer je praviloma nosilec vseh funkcij projekta in odgovoren za njegovo izvedbo. Newel (2020) pravi, da je projektni menedžer odgovoren skoraj za vse. Preprosteje bi bilo opredeliti, za kaj ni odgovoren. Vloga projektnega menedžerja je pri projektu najpomembnejša in mora imeti popolno podporo pri vodilnem menedžmentu (Rozman & Stare, 2008).

Projektni menedžer mora imeti širok spekter znanja in izkušenj. Nadzirati mora veliko področij, ki jih pokrivajo različni strokovnjaki. Od njega se pričakuje, da bo vse naloge uspešno povezal med seboj in sestavil povezano, usklajeno delujočo enoto (Rozman & Stare, 2008).

Kompleksnost vodenja projekta je večja tudi, ker je projektni tim sestavljen v kratkem obdobju. Člani ekipe se morda med seboj ne poznajo, kar lahko vpliva na projekt. Zgodi se, da se tudi med izvedbo projekta člani tima menjajo. Projektni menedžer mora ustvariti dobre poslovne in osebne odnose v ekipi (Rozman & Stare, 2008).

V nadaljevanju je naštetih nekaj ključnih nalog projektnega menedžerja. Pomembno je, da se zna odločati, načrtovati in kontrolirati.

- **Odločanje**

Naloga projektnega menedžerja je odločanje. Aktivnosti so običajno prepletene med seboj. Potrebno je odločanje o dejavnostih, ki jih bodo člani tima izvajali (Rozman & Stare, 2008).

– **Načrtovanje**

Projektni menedžer mora načrtovati cilje in definirati namen projekta s pomočjo naročnika. Sledi načrtovanje organizacije. Treba je postaviti roke zaposlenih, sredstev in stroškov, zadolžitve in odgovornosti (Rozman & Stare, 2008).

– **Kontrola**

Projektni menedžer mora neprestano nadzirati napredek projekta, ugotavljati, ali projekt sledi zastavljenim ciljem. Sem spada nadzor nas doseganjem rokov, nadzor nad zaposlenimi, nadzor nas sredstvi ter stroški (Rozman & Stare, 2008).

2 METODOLOGIJE PROJEKTNEGA MENEDŽMENTA

Projekt se lahko vodi na več načinov in z več metodami. Projektni menedžer mora izbrati ter določiti najprimernejšo metodologijo vodenja projekta.

2.1 Tradicionalne metode

V preteklosti je bilo razvijanje programske opreme preprostejše. Programi niso bili tako zahtevni in niso dosegali takih razsežnosti. Veljalo je pravilo »piši in popravi« (angl. code and fix). Tak način je še vedno v rabi pri razvoju manj zahtevne programske opreme v srednjih in manjših podjetjih. S tehnološkim napredkom se je zahtevnost razvijanja programske opreme znatno povečala. Pojavila se je potreba po kompleksnejših programih, kar lahko počnejo timi pod vodstvom menedžmenta.

S tem namenom so se v začetku sedemdesetih let začele razvijati tradicionalne metode, ki zdaj obstajajo že več kot 50 let. Te metode so predstavniki linearnega pristopa. Najbolj reprezentativna metoda tradicionalnega modela je slapovni model, katerega začetki segajo v leto 1970. To, da so tradicionalne metode zastarele, ni glavni problem, razlogi se skrivajo drugje.

Klasične metode temeljijo na življenjskem ciklu. Najpogosteje je izbran način vodenja programskih rešitev z zasnovo modela slapov. Ta koncept sledi prepričanju, da mora biti vsaka faza projekta izvedena zaporedoma in šele takrat, ko je prejšnja faza končana.

Naloga projektnega menedžerja pri klasičnih metodah je, da pripravi načrt in urnik izvajanja del ter nadziranje procesa. Tradicionalne metode zahtevajo natančen opis procesa, kar naj bi pripeljalo do lažje izvedbe. Nadziranje procesa se izvaja s popravki, načrt pa ne dopušča uvajanja novih stvari, ki bi lahko izboljšale proces in projekt. Iz tega je sledilo prepričanje, da lahko vsak dober projektni menedžer vodi katerikoli projekt razvijanja programske opreme ne glede na to, ali se spozna na vsebino. Najpogosteje opažene težave ob razvijanju programske opreme s pomočjo tradicionalnih metod so podane v nadaljevanju.

Motivacija ne prihaja od znotraj. Projekt se že od samega začetka načrtuje tako, da zadovolji zahteve strank oziroma naročnikov. Projektnemu menedžerju se ne dopusti samoiniciativnost.

Način vodenja projektov s pomočjo slapovnega modela temelji na temeljih proizvodnih in gradbenih industrij. Za te industrije je značilno, da so spremembe ali prilagoditve projektov lahko zelo drage. Ob povečanju potreb po razvijanju programske opreme je ta model postal najbolj priljubljen.

Prvo predstavitev, ki je opisovala uporabo takšnega načina razmišljanja v svetu programske opreme, je 29. junija 1956 organiziral Herbert D. Benington na simpoziju o naprednih programiranju metod za digitalne računalnike. Predstavitev je govorila o razvoju programske opreme za SAGE.

Leta 1983 je bil prispevek objavljen s predgovorom Beningtona, v katerem je pojasnil, da so bile faze namenoma organizirane v skladu s specializacijo nalog, in poudaril, da postopek dejansko ni potekal strogo od zgoraj navzdol, ampak je bil odvisen od prototipa.

Prvi formalni opis slapovnega modela je naveden članek iz leta 1970 Winstona W. Royceja, vendar Royce v tem članku ni uporabil izraza slapovni model.

Najzgodnejša uporaba izraza »slap« je bila morda objavljena v dokumentu iz leta 1976, ki sta ga objavila Bell in Thayer.

Leta 1985 je ministrstvo za obrambo Združenih držav Amerike (v nadaljevanju ZDA ta pristop zajelo v dokumentu DOD-STD-2167A, njihovih standardih za delo z izvajalci razvoja programske opreme, ki je dejalo, da bo izvajalec izvedel cikel razvoja programske opreme, ki vključuje naslednjih šest faz: predhodni načrt, podrobno načrtovanje, kodiranje in preskušanje enot, integracijo in testiranje. Motivacija za načrtovanje je pogosto bolj rezultat želje po kontroli kot pa želje po uspešno izvedenem projektu.

– **Načrtovanje in kontrola sta osrednji točki organizacije**

Tradicionalne metode se delijo na dve glavni fazi: analizo in spoznavanje. Če želimo dobiti uporabno programsko opremo, je treba ti dve glavni fazi razdeliti na: sistemske zahteve, programske zahteve, programiranje in dejansko delovanje. Osnova tradicionalnih metod je, da so vsa pričakovanja in cilji projekta definirani na samem začetku projekta, sledi postavitev načrta razvoja, temu sledi implementacija po zastavljenem načrtu**.** Iz tega lahko razberemo, da lahko šele ob zaključku projekta ugotovimo, ali smo zadovoljili potrebe. Lahko rečemo, da s tem ni nič narobe, ker smo sledili načrtu. Z načrtom, ki smo ga definirali na samem začetku, ne bomo zajeli sprememb, ki so se odvijale med razvijanjem projekta. Zavedati se moramo, da projekti lahko trajajo več let.

Tabela 1: Primerjava tradicionalnih in agilnih pristopov

Karakteristike	**Tradicionalne metode**	**Agilne metode**
Osnovne predpostavke in cilji	Plan je za razvijanje produkta je podrobno in natančno opredeljen. Vse zahteve so zapisane in predvidene vnaprej. Cilj je doseganje predvidljivosti in optimizacije procesa. Tradicionalne metode niso nagnjene k spremembam.	Kakovostna in prilagodljiva programska rešitev je lahko razvita z razmeroma majhnim timom, uporablja princip iterativnega razvoja, popravkov in rednih testov, ki dajejo hitre povratne informacije za učinkovite spremembe. Agilen pristop pričakuje poznavanje procesa za zagotavljanje hitrosti.
Dokumentacija	Veliko dokumentacije, zahtevajo se eksplicitna znanja.	Lahka, nadomesti se z osebno komunikacijo, tiho znanje.
Vodenje	Procesno orientirano.	Usmerjeno v ljudi.
Stil vodenja	Ukazovanje in nadziranje.	Vodenje in sodelovanje.
Upravljanje z znanjem	Eksplicitno.	Tiho.
Razvojni tim	Individualna vloga – želena specializacija, plansko orientirani, strukturirani vnaprej.	Samo organizacijski timi – spodbujanje menjave vlog, ko so locirani sodelujoči.
Komunikacija	Formalna.	Neformalna.
Vloga naročnika	Pomembna, majhna in pasivna soudeležba v projektih.	Ključna, aktivna, naročnik velja za del razvojnega tima.
Razvojni stil	Predvidevanje.	Prilagodljiv.
Potek projekta	Usmerjen po nalogah in aktivnostih.	Usmerjen glede na funkcije programske rešitve.
Razvojni modeli	Življenjski cikel (slapovi, spiralni).	Evolucijsko-dostavni model.
Merjenje uspeha	Skladnost z načrtom.	Podana poslovna vrednost.
Zaželena organizacijska oblika/struktura	Mehanistična (birokratska z visoko formalizacijo).	Organska (fleksibilna, spodbuja sodelovanje in kooperativne socialne akcije).
Tehnologija	Ni omejitev.	Prednost imajo objektno orientirane tehnologije.
Zahteve uporabnika	Neprestano sodelovanje z uporabnikom produkta.	Definirane pred začetkom podjetja.
Vključenost klientov	Visoko.	Nizko.
Razvojni model	Evolucijski.	Življenjski cikel.
Vključenost uporabnikov	Uporabniki sodelujejo čez celotni proces razvijanja projekta.	Uporabniki sodelujejo samo ob začetku razvijanja projekta.
Upravljanje z eskalacijami	Celotna ekipa sodeluje pri reševanju problema.	Eskalacije k višjemu menedžmentu.
Preference modela	Agilni model daje prednost prilagajanju.	Tradicionalni model daje prednost pričakovanju.

se nadaljuje

Tabela 1: Primerjava tradicionalnih in agilnih pristopov (nad.)

Produkt/proces	Manj osredotočenosti na procese.	Več osredotočenosti na procese.
Ocenjevanje napora	SCRUM-master olajša proces in ekipa naredi oceno.	Projektni menedžer poda oceno ter dobi potrditev.
Pregledi in odobritve	Pregledi se opravijo po vsaki ponovitvi.	Odvečni pregledi ter odobritve s strani vodij.

Vir: lastno delo.

2.2 Agilne metodologije

2.2.1 Kaj je agilnost?

Razvijanje programske opreme je bilo veliko časa dominirano s strani paradigme industrijskih pogledov in prepričanj. Metoda je temeljila na starih rutinah in teorijah, izoblikovanih v preteklosti. Veljalo je prepričanje, da zaposlenemu ne dajemo možnosti samostojnega, inteligentnega, avtonomnega odločanja. Veljalo naj bi, da lahko zaposleni opravlja le natančno definirane naloge. Delo, ki ga zaposleni opravlja, mora pripraviti bolj izkušen ali višje ranžirani zaposleni. S tradicionalnimi metodami naj bi stalno nadzirali zaposlenega in poskrbeli, da sledi definiranemu procesu dela. S tem ustvarimo dodatno delo menedžerju in hkrati ustvarimo občutek manjvrednosti pri zaposlenem. Kakovost zagotavljamo s sprejemanjem dobrih in zavračanjem slabih rezultatov. Monetarne nagrade so uporabljene za stimulacijo želenega obnašanja. Uporabimo strategijo korenčka in palice (Verheyen, 2019).

Pomanjkljivosti tradicionalnega pogleda pri razvijanju kompleksne programske opreme so zabeležene ter dokazane. Raziskava, ki jo je opravilo podjetje The Standish Group International, Inc. (2012), dokazuje nizko uspešnost razvijanja programske opreme, ki temelji na tradicionalnih metodah. Z uporabo tradicionalnih metod so podjetja uspešno razvila zgolj 10–20 % kompleksne programske opreme. Zaradi poraznih rezultatov je sledilo znatno znižanje pričakovanj, da bo neki projekt uspešno izpeljan. V industrijski paradigmi je bila definicija uspeha zelo preprosta. Cilj je bil zgolj, da so bile razvite prej definirane funkcionalnosti v začrtanem časovnem obdobju in v okvirih budžeta. S tem ko se podjetja niso prilagajala zahtevam na trgu, se je velikokrat razvijala funkcionalnost, ki je stranke niso nikoli uporabile. Neuporabljenih naj bi bilo več kot 50 % funkcionalnosti (Verheyen, 2019).

Industrijska paradigma je tako spravila industrijo razvijanja programske opreme na kolena. Sledilo je reševanje težav z ustvarjanjem še več načrtov, pripravljanjem dela za prihodnost, vendar to ni rešilo problema, ampak ga zgolj poglobilo. Še vedno je veljalo, da delavec ni sposoben opravljati dela sam (Verheyen, 2019).

V začetku novega tisočletja se je začela pojavljati beseda agilnost. Zgodila se je točka preobrata v industriji razvijanja programske opreme. Nova paradigma je tako vrnila upanje v industrijo razvijanja programske opreme. Razmišljanje, da delavec ne more opravljati sam s svojo glavo, se je tako začelo spreminjati (Verheyen, 2019).

Iz tega lahko sklepamo, da je preskok na agilno paradigmo skoraj nujen in industriji razvijanja programske opreme prinese veliko prednosti. Opravljene so bile tudi različne raziskave. Ena izmed teh je tudi raziskava »The chaos report«, opravljena s strani The Standish Group International, Inc. (2012). V raziskavi so bili podrobno analizirani projekti, kjer so bile vpeljane agilne metode in projekti, razviti po tradicionalnih metodah. Dokazano je bilo, da so bili projekti, ki so bili izvedeni po agilni metodi proti tradicionalnim kvalitetneje izvedeni, in to kar za 300 % (Verheyen, 2019).

2.3 Definicija agilnosti

Agilnost lahko opišemo kot 3 ključne karakteristike:

1. ljudje upravljajo proces;
2. iterativno-inkrementalni proces;
3. vrednost je merilo uspeha.

- **Ljudje upravljajo s procesom**

Tradicionalno je bil načrt razvijanja programske opreme detajlno definiran in zasidran. Agilnost daje moč in zahteva od zaposlenega, da se načrt razvijanja konstantno nadgrajuje. Ni proces prelaganja delavnih nalog drugim oddelkom, kjer vsak oddelek izvede specifično delo, ampak je gnana z neprestanim sodelovanjem med sodelavci vseh oddelkov. Agilnost ne prepoznava tradicionalnega pogleda »businessa« proti oddelku za informacijsko tehnologijo (Verheyen, 2019).

Za uspešno izvajanje agilne paradigme je potreben drugačen slog vodenja. Omogočena je, ko se vodstvo prilagaja delavcem. Naloga vodstva je, da postavi cilje ter usmerja ekipo v pravo smer. Tako se lahko zaposleni sami organizirajo in tako postanejo svobodnejši pri izražanju svoje ustvarjalnosti in inteligentnosti. Ustvari se tudi vzdržna hitrost izvajanja del (Verheyen, 2019).

- **Iterativno-inkrementalni proces**

Procesi agilnega razvijanja so definirani in zahtevajo visoko mero discipline. Produkt je sestavljen iz več kosov ali »inkrementov«. Vsak kos ali »inkrement« je lahko razširitev, izboljšava, modifikacija ali celo eliminacija neke funkcionalnosti. Sestavni deli programske opreme so običajno večkrat spremenjeni ali dograjeni za izboljšavo končnega proizvoda. Iterativno-inkrementalni proces je potreben, saj zahteve ali funkcionalnosti niso nikoli natančne in ne odražajo potreb, ki se pojavijo na trgu. Praviloma se te zahteve sproti

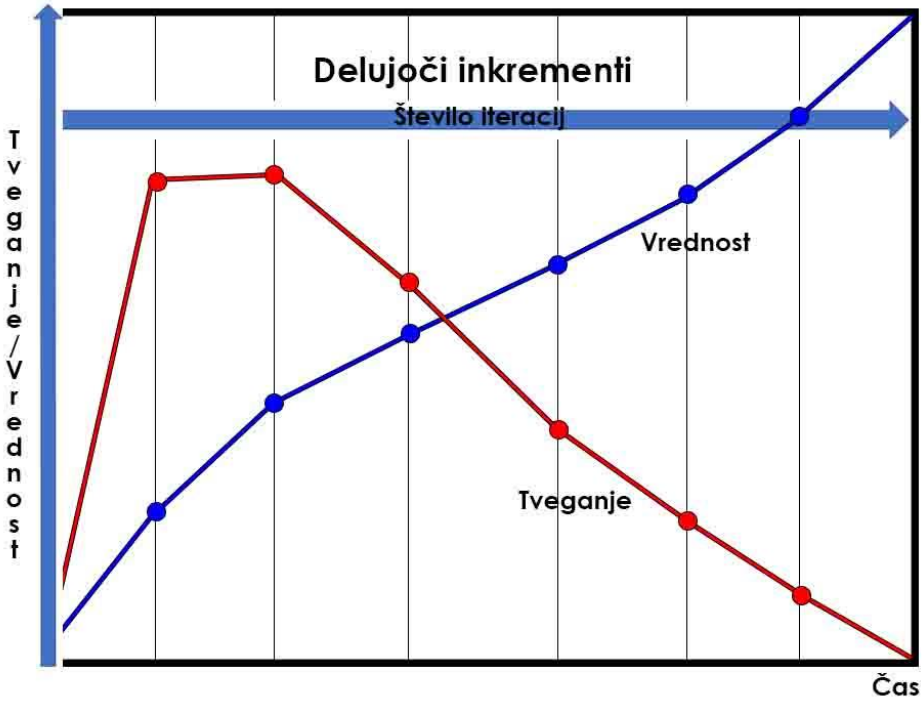
spreminjajo. Agilnost nam dovoljuje, da se na te potrebe ali zahteve pravočasno odzovemo. Če povzamemo, agilnost spodbuja spremembe kot vir za inovativnost in izboljšave (Verheyen, 2019).

– **Ustvarjena vrednost za stranko je merilo za uspeh**

Napredek in razvoj programske opreme po agilni metodi težko merimo s tradicionalnimi merili uspešnosti. Tradicionalno se uspešnost meri s primerjavo trenutnega stanja programske opreme z nekim prej definiranim dokumentom, ki ponazarja želeno točko v prihodnosti. Pri agilni metodi je zadeva drugačna. Programsko opremo sestavljamo po manjših delih ali »inkrementih«. Uspešni smo, ko je stranka zadovoljna in uporablja naš produkt. Pravilno je, da se pri razvijanju programske opreme obrnemo na uporabnika, cilj je potešitev njegovih potreb. S tem ko sproti nadgrajujemo produkt in ga stranka lahko testira, dobimo najbolj čisto sliko o uporabnosti našega produkta (Verheyen, 2019).

Pri agilnem pristopu določimo, na koliko časa bomo kos ali »inkrement« dodali k že obstoječi programski opremi. Pravilno je, da te časovne omejitve držimo. To nam omogoča, da smo osredotočeni in delamo tiste stvari, ki si jih zastavimo. V tej časovni periodi moramo zastavljene naloge izvesti, in če nam to ne uspe, lahko ugotovimo tudi razloge za neuspeh. Ko definiramo to časovno obdobje, pridejo na vrsto ponovitve (Verheyen, 2019).

Slika 1: Prikaz razvijanja z inkrementi



Vir: lastno delo.

S slike 1 je razvidno, da s ponovitvami povečamo vrednost za uporabnika in zmanjšamo tveganje za neuspeh projekta. Če vzamemo kot primer celotno industrijo informacijske

tehnologije, se vedno porajajo vprašanja, ali bo sploh sistem premogel nove zahteve. Bo sistem sposoben prenesti nove razširitve? Z agilnostjo se lahko izognemo tem težavam, saj se na potrebe sprotno odzivamo (Verheyen, 2019).

Razvoj nove programske opremo bi v zakup moral vzeti nepredvidene spremembe na trgu, razne zamude in nezadovoljstva strank. Podjetje se mora nujno odzvati. Tradicionalne metode teh sprememb ne predvidevajo.

Z agilnimi metodami se temu izognemo tako, da so nujne spremembe takoj naslovljene. Spremembe končnega produkta so zato hitre in učinkovite.

Agilnosti ni mogoče načrtovati. Je mešanica odzivnosti, hitrosti in prilagajanja. To je stanje, ki je potrebno za preživetje v tem nepredvidljivem okolju. Je brezpomenska, če se ne odraža v celotnem podjetju. Treba jo je vpeljati v odnose s kupci in trgi. S sprejetjem agilnosti se ustvarijo novi procesi, nova organizacijska kultura, ki temelji na učenju, neprestanih izboljšavah in inovativnosti (Verheyen, 2019).

Agilnosti:

– ni možno planirati;
– ni možno narekovati;
– ni možno kopirati;
– ni končna.

2.4 SCRUM

Tabela 2: Razlaga tujk uporabljenih v nadaljevanju naloge

Izraz metodologije SCRUM	**Slovenski prevod**
Daily Scrum	vsakodnevni sestanek za pregled poteka del na projektu
Product Backlog	seznam zahtev (množica uporabniških zgodb)
Product Owner	produktni vodja
Release Plan	plan izdaje
Scrum Team	razvojna skupina
Scrum Master	skrbnik metodologije
Sprint	iteracija
Sprint Backlog	seznam nalog, potrebnih za realizacijo posameznih uporabniških zgodb
Sprint planning	sestanek za načrtovanje iteracije
Sprint review & retrospective	sestanek za oceno kakovosti razvojnega procesa
User story	uporabniška zgodba
Velocity	hitrost razvoja (število točk, ki jih razvojna skupina lahko realizira v eni iteraciji)

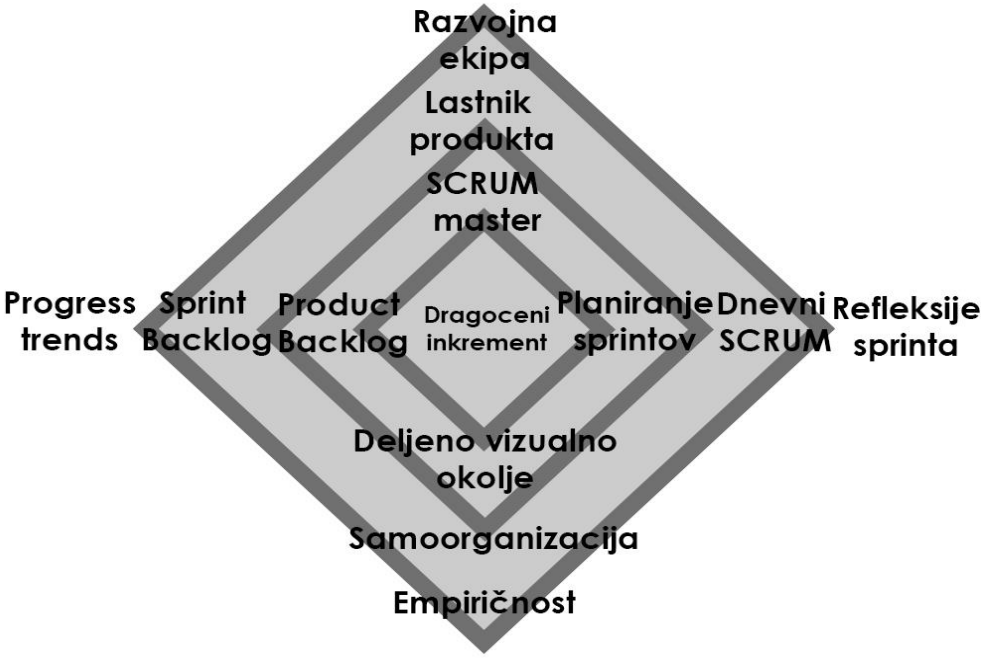
Vir: lastno delo.

2.4.1 Izvor SCRUM-a

Besedo »Scrum« sta prvič uporabila Takeuchi in Nonaka (1986) v članku »The New New Product Development Game«. Uspelo jima je dokazati, da so manjše ekipe uspešnejše pri razvijanju novih zahtevnih produktov. Najuspešnejše ekipe so prejele usmeritev. Pričakovano je bilo, da bodo same razvile svojo taktiko, kako najučinkoviteje rešiti problem. Ekipa potrebuje avtonomnost za dosego odličnosti. To sta ponazorila s primerom ekipe ragbija (Verheyen, 2019).

Beseda SCRUM je bila v kontekstu agilnosti prvič uporabljan s strani Jeffa Sutherlanda in Kena Schwaberja. SCRUM sta prvič izpostavila na konferenci Oopsla v Texasu, ZDA. Uporabila sta zasnovo iz članka in tako predstavila proces, namenjen izdelovanju kompleksne programske opreme. Trdita, da če članom ekipe naložiš naloge, ki potrebujejo zgolj izvedbo, se ti ne morejo razvijati in tako se jim lahko kompetence tudi zmanjšajo. Tako ne spodbujamo iskanja boljših ali učinkovitejših rešitev. Izgubi se tudi odprtost za boljše rešitve. SCRUM je »razvojni okvir« in ne metoda, je ogrodje, ki je bilo razvito za namen optimizacije in nadzorovanja razvijanja kompleksne programske opreme v nestabilnem okolju, organizacijah, poslu ali razmerah na trgu (Verheyen, 2019).

Slika 2: Razvojni okvir SCRUM-a



Vir: lastno delo.

Na sliki 2 lahko vidimo ključne elemente SCRUM-procesa. Ta zahteva visoko mero discipline, hkrati pa dopušča kreativnost in prilagajanje taktike. SCRUM temelji na medsebojnem spoštovanju in zahteva odgovornost.

Agilne metode temeljijo na principu oportunizma v poslu. Koncept razporejanja časa v krajša obdobja omogoča članom ekipe hitre odgovore na priložnosti, ki se pojavijo in prilagajanje na spremembe.

2.4.2 Vloge v procesu SCRUM-a

Ekipa je sestavljena iz:

- Product ownerja,
- razvojne ekipe,
- Scrum Masterja.

Vloga Product ownerja je, da v proces razvijanja izdelka prispeva znanje iz poslovnega vidika. Product owner v procesu razvijanja predstavlja vse deležnike, ki so lahko notranji in zunanji. Njegova naloga je posodabljanje in urejanje Product Backloga. Ta temelji na viziji končnega izdelka. Poskrbeti mora za dolgoročno vizijo izdelka in opravičiti obstoj tega izdelka.

Product Backlog prikazuje vse, kar si je Product owner zamislil. To so lahko funkcionalnosti, izboljšave, popravki, nadgradnje, ideje ali druge zahteve. Če si nekdo želi vedeti, kaj ekipa razvija, si to lahko pogleda v Product Backlogu.

Product owner pretvori pričakovanja s strani posla in ideje vpiše v Product Backlog. Potrebno pa je, da določi prednostne naloge. Njegova naloga je, da z omejenimi finančnimi sredstvi razvije ključne ali najpomembnejše funkcionalnosti in tako zadovolji uporabnika. Poskrbeti mora tudi za odnose z vsemi preostalimi deležniki.

- **Razvojna ekipa**

Razvojna ekipa se samoorganizira za najučinkovitejšo izvedbo vseh aktivnosti, ki so potrebne, da se stvari iz Product Backloga, ki jih je definiral Product owner v delujoči končni produkt. Razvojna ekipa mora poskrbeti, da so izvedene vse aktivnosti, ki so bile definirane v »sprintu«. To lahko vključuje kreiranje raznih testov, testiranj, programiranja, pisanja dokumentacije, integracij itd. Pokriva vse delo, ki mora biti opravljeno, da je kos ali inkrement na koncu sprinta delujoč in uporaben. Ob zaključku sprinta morajo biti zastavljene naloge dokončane in dodelane do take stopnje, da lahko izdelek roma v roke uporabnika. Zelo pomembno je, da se definira, kdaj je neki element dokončan. Razvojna ekipa mora postaviti tudi razvojne standarde, ki opisujejo, kako se izvedejo implementacije in vse sorodne dejavnosti. S tem zagotovimo potrebno kakovost za redno lansiranje inkrementov ali kosov k dotedanjemu produktu.

Razvojna ekipa oceni potrebno delo ali čas za vsak element v »Product Backlogu«. Razvojna ekipa izbere količino dela, ki ga lahko izvede v sprintu. Z oceno potrebnega dela lahko

predvidimo, koliko časa je potrebnega za izvedbo določene aktivnosti. Pridobljeno znanje pa lahko uporabimo v prihodnjih sprintih.

– **Scrum Master**

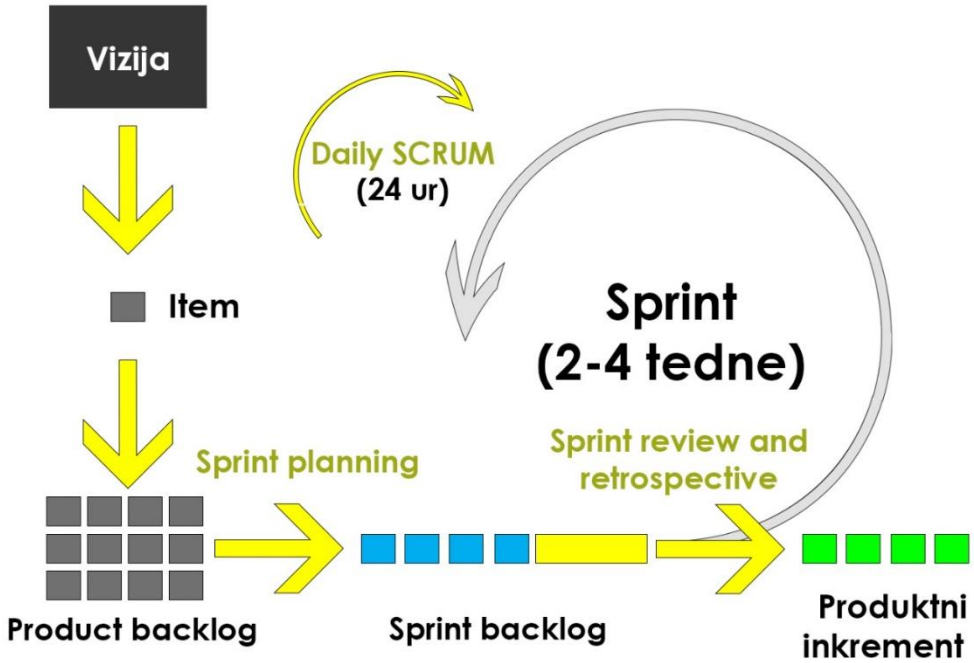
Naloga Scrum Masterja je, da povezuje produktnega lastnika in razvojno ekipo. Poznati mora delovanje koncepta SCRUM-a, učiti ekipo in organizacijo, da razumejo, spoštujejo in poznajo pravila SCRUM-a. Ena izmed njegovih nalog je, da poskrbi za odstranitev vseh ovir ali impedimentov v procesu.

– **Čas**

Časovne iteracije v procesu SCRUM so imenovane sprinti. Sprinti omogočajo razvojni ekipi usmeritev na doseganje cilja »sprinta« z najmanjšimi možnimi zunanjimi motnjami.

Vse delo v SCRUM-procesu je organizirano v sprinte. Cilj vsakega sprinta je, da k trenutnemu produktu doda uporabni del ali inkrement. Sprint običajno traja največ 4 tedne, običajno pa od 2 do 4 tedne.

Slika 3: Prikaz delovanja SCRUM-a



Vir: lastno delo.

V časovnem obdobju, ki ga imenujemo »sprint«, izvajamo dodatne aktivnosti za izboljšavo procesa in prilagajanje na nepredvidene spremembe. Te aktivnosti so:

1. Sprint planning,
2. Daily SCRUM,

3. Sprint review,
4. Sprint retrospective.

Vsak sprint se začne s Sprint planningom, kjer se razvojna ekipa in Product Owner sestanejo. Sestanek je namenjen določitvi elementov, ki jih bodo razvijali v prihajajočem sprintu. Ekipa pregleda Product Backlog in s pomočjo Product Ownerja določi prednostne elemente. S pomočjo izkušenj iz prejšnjih sprintov in z oceno zahtevnosti vsakega elementa si določi optimalno količino dela. Količino dela, ki je bila določena in locirana v Sprint Backlogu, mora biti dokončana do zaključka sprinta. Zgodi se tudi, da so bile podane ocene o zahtevnosti nekaterih elementov napačne, kar lahko onemogoči doseganje zastavljenega cilja. Sčasoma se določitev količine dela izboljša. Ustvarjeni produktni inkrement mora biti v takem stanju, da ga lahko priključimo k že obstoječem produktu (Verheyen, 2019).

Količino dela, ki ga opravimo v sprintu, imenujemo Velocity. Je kazalnik količine dela, ki ga je ekipa opravila v preteklem sprintu. Velocity je seštevek enot (denarnih ali dela). Metrika nam pove, koliko dela lahko planiramo v sprintu.

Količina izbranega ali Sprint Backloga je oblikovana, analizirana in definirana. Planiranje sprinta običajno traja manj kot 8 ur. Ko se faza planiranja zaključi, lahko razvojna ekipa začne delo (Verheyen, 2019).

Za upravljanje in sledenje poteku razvijanja, ki ga izvaja razvojna ekipa, organiziramo 15-minutni dnevni sestanek, ki ga imenujemo Daily SCRUM. Prisotnost Product ownerja ni obvezna, je pa zaželena. Na tem sestanku se razvojna ekipa pogovori o možnih problemih in prejme usmeritve s strani Product ownerja. Glavni namen sestanka je, da se v procesu poišče prostor za optimizacijo in olajša doseganje zastavljenega cilja. Vse dogovorjene spremembe so zabeležene v Product Backlogu (Verheyen, 2019).

Med izvajanjem sprinta se začnejo pojavljati inkrementi produkta. Če se lastniku produkta zdi inkrement uporaben, ga lahko lansira brez težav. Ob koncu sprinta se inkrement pregleda na Sprint reviewju. Na tem sestanku se pregleda podrobno delovanje funkcionalnosti in oceni, ali je inkrement dovolj dodelan za lansiranje (Verheyen, 2019).

Na Sprint reviewju Product owner pregleda dosedanje delo. Med pregledovanjem inkrementov ekipa preveri možne spremembe, ideje, izboljšave in pridobi povratne informacije o opravljenem delu. Dorečene spremembe so vnesene v Product Backlogu za kasnejšo implementacijo. Sprint review običajno traja do maksimalno 4 ure (Verheyen, 2019).

Sprint zaključimo s Sprint retrospectivom, kjer ekipa pregleda, reflektira delovni proces. Na tem sestanku se ekipa pogovori o vseh področjih dela. Pogovor poteka o samem produktu, uporabljeni tehnologiji, družbenih pogledih, SCRUM-procesu, razvojnih praksah, sodelovanju, kakovosti produkta itd. Namen sestanka je pogovor o tem, kaj v procesu deluje in kje se proces lahko izboljša. Sestanek traja maksimalno 3 ure (Verheyen, 2019).

– **Definicija zaključenega**

V definiciji zaključenega so izražene kakovosti in merila, ki morajo biti izpolnjeni, če želimo novo ustvarjeni inkrement lansirati v produkcijsko okolje. Definicija zaključenega se nanaša na kakovosti, aktivnosti, merila, naloge in delo, ki morajo biti izvedeni na delujočem inkrementu, da je ta produkcijske kvalitete. Definicija zaključenega je ključna za razumevanje in načrtovanje dela, potrebnega za ustvarjanje produkta v takem stanju, da ga lahko lansiramo. Definicija dokončanega omogoča transparentnost. Če je definirano, kaj pomeni, da je neki element zaključen, smo lahko prepričani, da bo dogovorjeno delo tudi dokončano.

3 EMPIRIČNI DEL

3.1 Uvodni del

V današnjih časih se podjetja soočajo z nestabilnim poslovnim okoljem in nenehnimi spremembami, pa tudi z vedno večjo konkurenco. Podjetja, ki delujejo v informacijski tehnologiji, so se morala prilagoditi z razvijanjem novih metodologij in načinov razvijanja programske opreme. Spoznala so, da je za uspešnost potrebno hitro prilagajanje na okolje. S tem se je rodila agilnost, ki se je v tem negotovem okolju izkazala za pravo pot, ključno za uspeh.

Negotovo okolje je tako pripeljalo do razvijanja in vpeljevanja novih metod vodenja projektov. Trenutno obstaja veliko metodologij, ki spadajo pod okrilje agilnih metod. Pojav agilnih metodologij se je začel v projektnem vodenju programske opreme, zdaj pa se pojavlja tudi pri razvijanju fizičnih produktov. Tradicionalne metodologije se soočajo z mnogimi problemi. Ugotovljena je prevelika obremenjenost zaposlenih z birokracijo, s težkim planiranjem, kar lahko vpliva tudi na zamujanje pri doseganju rokov, in nenatančnim planiranjem finančnih sredstev. Z agilnimi metodami lahko izboljšamo proces za vse deležnike.

Trenutno je ena izmed najbolj uporabljenih agilnih metodologij SCRUM razvojni okvir. SCRUM omogoča, da ekipa, ki sodeluje pri razvijanju programske opreme, neprestano izboljšuje samo sebe, proces in celotni digitalni produkt.

V sklopu diplomske naloge je bila uporabljena metoda polintervjuja, ki je obravnavala raziskovalno vprašanje:

Zakaj so pri razvoju programske opreme agilni pristopi uspešnejši v primerjavi s tradicionalnimi?

Svoje ugotovitve sem primerjal tudi z najdenimi publikacijami in raziskavami, ki obravnavajo to tematiko.

3.2 Analiza opravljenih intervjujev

V intervju sem zajel 10 ljudi, ki se ukvarjajo z razvojem programskih rešitev. Sodelujoči so zaposleni v srednjih in večjih podjetjih v Ljubljani. Intervjuje sem opravil s 4 projektnimi vodji ali lastniki produktov, 3 SCRUM-masterji in 3 programerji.

V intervjujih sem zastavil vprašanja, ki so bila vezana na oba načina razvijanja programske opreme. Zanimale so me prednosti in slabosti in želel sem odkriti, katera metodologija zagotavlja boljše rezultate za končnega uporabnika.

3.2.1 Agilne metodologije

Iz intervjujev sem ugotovil, da se projektni vodje odločajo za uporabo agilnih metod. Vsi vprašani uporabljajo razvojni okvir SCRUM. Uporabljen je zaradi poznavanja uspešnosti, ki je bila opažena pri drugih podjetjih. Projekti, vodeni s pomočjo agilnih metodologij, digitalnim projektom zagotavljajo kar 50 % večjo uspešnost.

3.2.1.1 Prednosti SCRUM razvojnega okvirja

– **Prednosti za lastnika produkta**

SCRUM razvojni okvir projektnemu vodji daje možnost lažjega in uspešnejšega prilagajanja razmeram na trgu, ker se lahko nemudoma odzovemo na vse spremembe, ki so opažene v mikro in makro okolju. Projektni vodja se lahko odziva tudi na potrebe, ki so zaznane znotraj podjetja in s strani vseh deležnikov.

Zagotovljena je nujno potrebna prilagodljivost, saj se potrebe končnega uporabnika neprestano spreminjajo. Z iterativnim inkrementalnim lansiranjem si zagotovimo neprestan pretok povratnih informacij. Projektni vodja lahko prilagodi razvojni načrt glede na potrebe, ki se pojavijo na trgu. Pridobi tudi možnost postavitve okvirnega končnega cilja in svobodo ob vmesnem času razvoja.

Lastnik produkta lahko produkt predstavi deležnikom že v začetnih fazah in ob vsakem inkrementu. To mu daje možnost, da ob najhitrejšem možnem času ugotovi, če gre razvoj v napačno smer. Hitro ugotavljanje napak izboljša hitrost in kakovost končnega produkta.

Lastnik produkta ima s SCRUM-procesom nadzor nad projektom. Sam razvoj je zelo transparenten, saj je voden z orodji, ki omogočajo nadzor nad delom, ki ga opravlja ekipa.

– **Prednosti za končnega uporabnika**

Ključ, ki omogoča SCRUM razvojnemu okvirju tako visoko uspešnost, je iterativno inkrementalno lansiranje produkta, ki omogoča, da se funkcionalnosti k programski opremi dodajajo postopoma. Končni uporabniki lahko vidijo spremembe ob koncu vsakega

2-tedenskega cikla. Končnega uporabnika tako ne preobremenimo z novimi funkcionalnostmi in mu damo dovolj časa, da se o tej funkcionalnosti poduči.

– **Prednosti za vodstvo**

SCRUM razvojni okvir omogoča tudi transparentnost razvoja na vsakem koraku. Projekti so vodeni s pomočjo programskih rešitev, kot so Azure DevOps, Jira, Asana. Vodstvo projekta ima hiter in preprost dostop do programskih rešitev, s katerimi je program voden in tako lahko vedno preverijo, kje je neka funkcionalnost in kako napreduje projekt.

SCRUM razvojni okvir je tudi cenovno ugoden. Če razvijamo programsko opremo, namenjeno končnemu uporabniku, pri kateri ne moremo definirati vseh potrebnih funkcionalnosti ali te napačno definiramo, nas lahko to veliko stane. Prednost SCRUM-a je zagotovo delujoča programska oprema.

– **Prednosti za razvojno ekipo**

Prednosti za razvojno ekipo so, da s pomočjo iterativnih inkrementov takoj prejmejo informacije s strani končnih uporabnikov o potrebnih popravkih. Pregledovanje in popravljanje kode je zato preprostejše, ker se programerji laže spomnijo, kje so potrebni popravki, ker so te funkcionalnosti razvijali v bližnji preteklosti.

Opolnomočenje ekipe omogoča članom, da si med seboj sami razdelijo prej določene naloge. S tem se izboljšajo počutje, samoiniciativnost in zaupanje.

Ekipa je s projektnim vodjo neprestano v stiku tudi na dodatnih sestankih, kot so retrospektiva sprinta, planiranje sprinta. Omogoča vključenost pri določanju nalog in podajanje mnenj. S tem se produkt in ekipa sprotno izboljšujeta.

3.2.1.2 Slabosti SCRUM razvojnega okvirja

– **Slabosti za lastnika produkta**

Slabost za projektno vodjo je, da pri razvojnem načrtu težko natančno opredeli časovni načrt razvijanja projekta, če to zahteva vodstvo. Težave se lahko pojavijo, če ekipa ni samoiniciativna in se stežka prilagodi spremembam. Agilne metodologije ni možno aplicirati na vse projekte.

– **Slabosti za končnega uporabnika**

Slabost za končnega uporabnika je lahko prehitro lansiran inkrement. Lahko ni dovolj natančno definiran in vsebuje veliko hroščev ali napak.

– **Slabosti za vodstvo**

Slabosti so težje predvidevanje stroškov projekta in manjši nadzor nad projektom. Funkcionalnosti so zgolj okvirno definirane. Težavna je implementacija SCRUM-procesa, ker zahteva menjavo mentalitete. Projekt morda nima konca. Projekt je lahko ob koncu videti popolnoma drugače, kot je bilo v samem začetku predvideno.

– **Slabosti za razvojno ekipo**

Potrebna je menjava mentalitete in miselnosti. Delo ni natančno opredeljeno za daljše obdobje. Pojavi se lahko potreba po menjavi nalog, definiranih v sprintu, kar lahko poruši načrt ekipe. Zaradi časovne stiske in hitenja lahko razvojna ekipa išče krajšnice, funkcionalnosti ne razvije natančno in kvalitetno in si za razvoj ne vzame dovolj časa. Razvojna ekipa lahko porabi veliko časa za samo komunikacijo.

3.2.2 Tradicionalne metodologije

Projektni vodje se za uporabo tradicionalnih metod ne odločajo več tako pogosto. Uporabi tradicionalnih metod slapovnega modela zdaj največkrat botruje narava projekta. Za uporabo tradicionalnih metod se odločajo, ko je projekt zelo jasno definiran. Kot primer je bil izpostavljen razvoj CRM-ja. Veliko projektnih vodij je izpostavilo, da so projekte pripeljali do določene stopnje in kasneje vpeljali agilne metode. Kot glavno prednost pri tradicionalnem razvoju projektni vodje poudarjajo stabilnost razvoja programske opreme. Stabilnost lahko opredelimo kot jasno opredeljene naloge vsakega člana ekipe in sledljivost samih nalog.

3.2.2.1 Prednosti »slapovnega modela«

– **Prednosti za projektnega vodjo**

Slapovni model omogoča projektnemu vodji lažje planiranje nalog za vse udeležence v procesu. Omogoča lažje sledenje razvoju. Pri tradicionalnih metodah je mogoče lažje tudi planiranje vseh ostalih aktivnosti po zaključenem projektu, kot je promocija.

– **Prednosti za končnega uporabnika**

Prednost za končnega uporabnika se pojavi, ko je projekt natančno definiran. Končni uporabnik lahko zelo hitro pride do končnega produkta.

– **Prednosti za vodstvo**

Pri razvijanju projektov s slapovnim modelom je v sam proces razvijanja programske opreme vključeno veliko predhodnega planiranja. Vodstvu je tako omočeno bolj enostavno planiranje stroškov projekta.

Vodstvu omogoča pregled nad celotnim projektom in celotnim časovnim načrtom razvijanja funkcionalnosti.

– **Prednosti za razvojno ekipo**

Prednosti za razvojno ekipo so, da so vse naloge natančno definirane in se od začetka do konca ne spreminjajo veliko. Vse funkcionalnosti so poznane, definirane in potrjene.

3.2.2.2 Slabosti tradicionalnih metod

– **Slabost za projektno vodjo**

Projektni vodja težko predvidi potrebe končnega kupca. Proces je tog in zakoreninjen. Med planiranjem in dokončanjem programske opreme lahko mine tudi več let. Lahko prihaja do ugibanja, kaj bo končni uporabnik dejansko potreboval.

Projektni vodja med samim razvojem ne more prilagajati razvojnega načrta in je omejen s plani, ki so bili začrtani na samem začetku projekta. Stežka predvidimo sprejetost programske opreme.

– **Slabosti za končnega uporabnika**

Končni uporabnik ob zaključenem razvoju programske opreme prejme rešitev, ki morda ne zadovolji njegovih pričakovanj.

– **Slabosti za vodstvo**

V projekt je bilo vloženega veliko denarja. Lahko se zgodi, da je programska rešitev neuporabljena. Če programska oprema za končnega uporabnika ni uporabna, so lahko spremembe zelo drage in časovno potratne.

– **Slabosti za razvojno ekipo**

Slabost za razvojno ekipo je natančno definiran rok, kdaj mora biti programska oprema dokončana. To lahko predstavlja velik stres za razvojno ekipo.

Stežka naslovimo hrošče in nujne popravke, saj so ti lahko umeščeni šele ob koncu planiranega razvoja. To predstavlja dodatno obremenitev za razvojno ekipo, saj je treba veliko dela s seznanjanjem z že spisano kodo. Spreminjanje neke funkcionalnosti lahko vpliva na druge funkcionalnosti in tako poruši nekatere že delujoče dele programske opreme.

Ko je programska oprema lansirana na tržišče, lahko pride do veliko želj po popravkih in je zato lahko ekipa zasuta z delom.

SKLEP

Če primerjamo agilno metodologijo SCRUM in tradicionalno metodo slapovni model ugotovimo, da imata oba pristopa prednosti in slabosti za različne deležnike.

– **Projektni vodja**

Agilna metodologija SCRUM omogoča projektnemu vodji, da programsko opremo pokaže deležnikom, preden so zaključene vse funkcionalnosti. Omogoča mu tudi, da končni uporabnik poda povratne informacije o dodani funkcionalnosti ob koncu vsakega sprinta. To mu omogoča, da hitro ugotovi, če gre razvoj programske opreme v napačno smer. Tako se izboljšata hitrost in kakovost končnega produkta. Tradicionalna metoda slapovni model tega ne omogoča.

Agilni razvojni okvir SCRUM projektnemu vodji omogoča neprestano prilagodljivost razvojnega načrta razmeram na trgu, saj od končnega uporabnika neprestano prejema povratne informacije, zato se lahko prilagaja končnemu uporabniku. Tradicionalni slapovni model tega ne omogoča. SCRUM-metoda omogoča tudi večjo transparentnost razvoja, saj je vsaka naloga natančno zabeležena.

Tradicionalni slapovni model omogoča natančno definicijo razvojnega načrta, česar agilna metodologija SCRUM ne omogoča. Pri slapovnem modelu lahko pride do veliko ugibanj, kar pripelje do napačne definicije razvojnega načrta, pri SCRUM-u se to zelo težko zgodi, saj lahko funkcionalnosti sproti prilagajamo. Napaka pri definiranju funkcionalnosti lahko pomeni veliko dodatnih stroškov in slabe volje pri vseh deležnikih.

Kot glavno prednost tradicionalnega slapovnega modela lahko izpostavimo hitrost, lažje načrtovanje aktivnosti po dokončanju razvoja. Vendar je pogoj, da smo vse funkcionalnosti pravilno predvideli.

Ugotovitve potrjuje tudi publikacija »Agile Software Project Management with Scrum« (Mahnic & Drnovscek, 2005) na primeru razvoja programske rešitve Univerze v Ljubljani. SCRUM-proces omogoča produktnemu lastniku večjo transparentnost nad projektom in lažje upravljanje projekta. Kot velika prednost je izpostavljeno tudi inkrementalno lansiranje, kar je omogočilo dodajanje delujočih funkcionalnosti na dvotedenski ravni.

Prednosti za projektnega vodjo potrjuje tudi publikacija »Benefits & Pitfalls of using Scrum software development methodology« (Adell, 2013). V njej so izpostavljene prednosti za projektnega vodjo, saj SCRUM omogoča, da so ključne funkcionalnosti programske opreme dokončane v najhitrejšem možnem času, omogoča večje prihranke proti tradicionalnim metodam, kar omogoča produktnemu vodji razvoj večfunkcionalnosti. Med samim razvojem mu omogoča prejemanje povratnih informacij in stik z razvojno ekipo ter lažje prilagajanje spremembam v okolju.

Lahko povzemamo, da je SCRUM primernejši za razvoj, saj je v trenutnem okolju težko predvideti vse spremembe in zahteve.

– **Končni uporabnik**

Pri agilni metodologiji SCRUM se neprestano zanašamo na povratne informacije, to pomeni, da lahko na dolgi rok zagotovimo najvišjo možno kakovost v primerjavi s tradicionalno slapovno metodo.

Končni uporabnik lahko prejme dokončano programsko opremo hitreje s pomočjo tradicionalne slapovne metode, vendar je lahko produkt zanj neuporaben, ker so bile funkcionalnosti napačno predvidene.

S SCRUM-om izločimo možnost neuporabnosti, saj je končni uporabnik vključen v proces. S sprotnim lansiranje lahko končnemu uporabniku dostavimo uporabno programsko opremo hitreje kot tradicionalni slapovni model. Programska oprema ne bo takoj zajemala vseh želenih funkcionalnosti. Imela bo le del želenih funkcionalnosti in končni uporabnik tako lahko poda mnenja za izboljšavo.

Prednost SCRUM-a je tudi ta, da končnega uporabnika ne zasujemo z vsemi funkcionalnostmi naenkrat. Tradicionalni slapovni model pa tega ne omogoča.

Publikacija »Agile project management with Scrum: A case study of a Brazilian pharmaceutical company IT project« pritrjuje, da so končni uporabniki bolj zadovoljni z dostavljeno programsko opremo, ki je bila razvita s SCRUM-metodo, saj je ta hitreje dostavljena in vsebuje vse želene funkcionalnosti.

V sedanjih časih, ko je okolje nepredvidljivo, je skoraj nemogoče predvideti vse funkcionalnosti natančno, zato je primernost SCRUM-a toliko večja.

– **Vodstvo podjetja**

Agilna metodologija SCRUM ne omogoča take transparentnosti financ, razvojnega načrta, predvidljivosti in nadzora, kot to omogoča tradicionalna slapovna metoda. Še vedno pa omogoča transparenten pregled nad planiranim delom.

Če si vodstvo želi zadovoljiti potrebe končnega uporabnika, kar je običajno želja vodstva, bo za razvoj uporabilo SCRUM-metodo, saj bo zadovoljstvo kupca in cena samega projekta nižja, torej se s finančnega vidika bolj splača agilna metoda SCRUM, saj zagotavlja, da bodo potrebe končnega kupca zagotovo zadoščene. Slapovni model vodstvu ne zagotavljajo uspešnosti.

Povečanje kvalitete produkta in večje zadovoljstvo vodstva s končnim produktom potrjuje tudi publikacija »Agile project management with Scrum: A case study of a Brazilian pharmaceutical company IT project« (Azanha, Argoud, Camargo Junior & Antoniolli,

2017), kjer je bilo ugotovljeno, da je bila kvaliteta dostavljenega produkta zelo velika in je programska oprema razvita s pomočjo SCRUM-metode organizacijo dodala visoko dodano vrednost. Kot prednost so izpostavili tudi 75-odstotno zmanjšanje časa, potrebnega za razvoj, kar je vplivalo tudi na manjše stroške končnega produkta.

V publikaciji »Benefits & Pitfalls of using Scrum software development methodology« (Adell, 2013) je prav tako potrjeno znižanje stroškov pri razvoju programske opreme, kjer je omenjeno, da je kljub nižji ceni dostavljena programska oprema kvalitetnejša. Omenjena je tudi slabost, da se spreminjata razvojni načrt in težji nadzor vodstva nad samim projektom.

– **Razvojna ekipa**

Razvojni ekipi SCRUM omogoča vključenost v projekte, česar slapovni model ne omogoča. Razvojna ekipa pri agilnem pristopu SCRUM nima tako jasno postavljenih nalog na dolgi rok kot pri tradicionalni metodi slapovni model. V SCRUM-u se naloge definirajo na 2-tedenske cikle, kar je lahko zahtevno za nekatere razvijalce. SCRUM-metode zahtevajo tudi več komunikacije razvojne ekipe in lastnika produkta, kar je lahko časovno potratno. Pri slapovnem modelu so funkcionalnosti natančno definirane. SCRUM razvojni okvir omogoča članom ekipe, da si sami razporejajo naloge in temelji na samoiniciativnosti. Tradicionalne metode temeljijo na strogo delegiranih nalogah s strani vodstva. SCRUM predvideva izboljšave produkta in tudi same ekipe, česar slapovni model ne vključuje.

Ugotovitev sta potrdila tudi Mahnic in Drnovscek (2005) v publikaciji »Agile Software Project Management with Scrum«, kjer je bilo na primeru razvoja informacijskega sistema Univerze v Ljubljani s pomočjo SCRUM-a potrjeno, da je komunikacija med člani ekipe boljša in je prispevala k večji meri sodelovanja. Povečala sta se tudi motivacija in odgovornost za uspeh projekta. Članom ekipe je SCRUM-proces omogočil več svobode za unovčenje svoje iznajdljivosti in znanja med vsakim sprintom.

LITERATURA IN VIRI

1. Adell, L. (2013). *Benefits & Pitfalls of using Scrum software development methodology* [objava na bogu]. Pridobljeno 8. maja 2020 iz https://www.belatrixsf.com/blog/wp-content/uploads/kalins-pdf/singles/benefits-pitfalls-of-using-scrum-software-development-methodology.pdf
2. Azanha, A., Argoud, A.R.T.T., Camargo Junior, J.B.D. & Antoniolli, P.D. (2017), "Agile project management with Scrum: A case study of a Brazilian pharmaceutical company IT project". *International Journal of Managing Projects in Business*, *10*(1), 121-142.
3. Cleland, I. D. (1999). *Project management - Strategic design and implemantation*. New York: McGraw-Hill.
4. Lewis, J. P. (1998). *Mastering Project Management- applying advanced concepts of systems thinking, control and evaluation, resource allocation*. New York: McGraw-Hill.

5. Mahnic, V. & Drnovscek, S. (2005). *Agile Software Project Management with Scrum.* Ljubljana: Fakulteta za računalništvo in informatiko.
6. Project Management Institute, Inc. (2004). *Project Management Body of Knowledge* (3. izd.). Newtown: Project Management Institute, Inc.
7. Rozman, R. & Stare, A. (2008). *Projektni management ali ravnateljevanje projekta.* Ljubljana: Ekonomska fakulteta.
8. Takeuchi, H. & Nonaka, I. (1986). The new new product development game. *Harvard business review*, *64*(1), 137–146.
9. The Standish Group International, Inc. (2012). *The CHAOS Manifesto 2012: The Year of the Executive Sponsor*. Boston: The Standish Group International, Inc.
10. Verheyen, G. (2019). *SCRUM A pocket Guide* (2. izd.). Hertogenbosch: Van Haren Publishing.