UNIVERZITET U SARAJEVU
EKONOMSKI FAKULTET

ZAVRŠNI RAD

METODE PROJEKTNOG MENADŽMENTA U IT INDUSTRIJI U BIH

Sarajevo, novembar 2023. LEMANA ĆENANOVIĆ

POSVETA

Sva hvala i zahvala pripada Uzvišenom Allahu, koji je u imperativu rekao „Ikre“, što znači „Čitaj“ i time nam u emanet ostavio da budemo od onih koji znaju, a ne od onih koji ne znaju. Hvala Mu na svakom stanju, na svakom uspjehu, jer bez Njegove volje ništa ne bi bilo ostvareno. Potom, hvala mojoj porodici koja me je podsticala na dobro i bila mi podrška na mom edukacijskom putu, a posebno hvala mojoj majci koja nije bila samo majka, već i najbolja prijateljica, sestra, učiteljica, doktorica, domaćica, moja ruka podrške, nježnosti i milosti. Hvala svakoj osobi koja je doprinijela u bilo kojem segmentu moga života, a tu je zasigurno veći broj ljudi. Hvala mom mentoru na susretljivosti, uputama u procesu izrade mog rada i na fleksibilnosti njegovog pristupa. Od srca vam se svima zahvaljujem uz dovu Uzvišenom Allaha da nam učini od koristi ono čemu nas je podučio i da nas poduči onome što će nam koristiti, te da nam poveća znanje. Molim Allaha da vas nagradi na oba svijeta. Amin.

U skladu sa članom 54. Pravila studiranja za I, II ciklus studija, integrisani, stručni i specijalistički studij na Univerzitetu u Sarajevu, daje se

IZJAVA O AUTENTIČNOSTI RADA

Ja, Lemana Ćenanović, studentica drugog (II) ciklusa studija, broj index-a 4433-72676 na programu Menadžment, smjer Menadžment i informacione tehnologije, izjavljujem da sam završni rad na temu:

METODE PROJEKTNOG MENADŽMENTA U IT INDUSTRIJI U BIH

pod mentorstvom prof. dr. Aziz Šunje, izradila samostalno i da se zasniva na rezultatima mog vlastitog istraživanja. Rad ne sadrži prethodno objavljene ili neobjavljene materijale drugih autora, osim onih koji su priznati navođenjem literature i drugih izvora informacija uključujući i alate umjetne inteligencije.

Ovom izjavom potvrđujem da sam za potrebe arhiviranja predala elektronsku verziju rada koja je istovjetna štampanoj verziji završnog rada.

Dozvoljavam objavu ličnih podataka vezanih za završetak studija (ime, prezime, datum i mjesto rođenja, datum odbrane rada, naslov rada) na web stranici i u publikacijama Univerziteta u Sarajevu i Ekonomskog fakulteta.

U skladu sa članom 34. 45. i 46. Zakona o autorskom i srodnim pravima (Službeni glasnik BiH, 63/10) dozvoljavam da gore navedeni završni rad bude trajno pohranjen u Institucionalnom repozitoriju Univerziteta u Sarajevu i Ekonomskog fakulteta i da javno bude dostupan svima.

Sarajevo, 31.10.2023. Potpis studenta/studentice:

SAŽETAK

Kako bi IT kompanije optimizirale svoju učinkovitost, one su podložne primjeni različitih strategija. Jedna od njih je upravo upotreba određenog metodološkog pristupa, odnosno metodologije projektnog menadžmenta, koja je sadržana od seta tehnika, praksi, metoda, pravila i postupaka. Ona se zasniva na konkretnom pristupu razvijanja projekta, te se s toga vrlo lahko dešava da način upravljanja poslovanjem varira među kompanijama shodno njihovim potrebama.

Istraživački rad predstavlja dvije osnovne metode projektnog menadžmenta, tradicionalnu i agilnu, uz njihovu komparativnu analizu, a nakon toga i njihove različite podmetodologije sa popratnim objašnjenjem njihovih karakteristika. U grupi podmetodologija obuhvaćen je temeljit pregled literature tradicionalnog vodopad i spiralnog modela, te također agilne metodologije poput Ekstremnog programiranja, Scrum-a, Kanban-a, te „Feature driven“, „Dinamic system“ i „Crystal“ razvojnih okvirova. Istraživanje sporovedeno u IT kompanijama imalo je za cilj da pruži rezultate koji će jasno naznačiti koja od dvije osnovne metodologije je najzastupljenija na prostoru Bosne i Hercegovine, zbog čega, te da li je agilna metodologija pod nazivom „Scrum“ najrasprostranjenija.

Ključne riječi: Tradicionalni projektni menadžment, Agilni projektni menadžment, Metodologija, Principi metodologije, Scrum

ABSTRACT

To optimize their efficiency, IT companies are susceptible to employing various strategies. One of them is the use of a specific methodological approach, namely project management methodology, which consists of a set of techniques, practices, methods, rules, and procedures. It is based on a particular project development approach, making it easy for management practices to vary among companies according to their needs.

This research paper explores two fundamental project management methods, traditional and agile, along with their comparative analysis, followed by a presentation of their respective sub-methodologies with accompanying explanations of their characteristics. In the group of sub-methodologies, a comprehensive review of literature covers the traditional Waterfall and Spiral models, as well as agile methodologies such as Extreme Programming, Scrum, Kanban, and development frameworks like „Feature Driven“, „Dynamic System“, and „Crystal“. The research conducted in IT companies aimed to provide results that clearly indicate which of the two primary methodologies is most prevalent in Bosnia and Herzegovina and whether the agile methodology known as "Scrum" is the most widespread.

Key words: Traditional project management, Agile project management, Methodology, Methodology principles, Scrum

SADRŽAJ

1. UVOD 1

1.1. Predmet istraživanja 3

1.2. Istraživačko pitanje 3

1.3. Cilj istraživanja 3

1.4. Hipoteze istraživanja 4

1.5. Metodologija 4

1.6. Struktura rada 5

2. PROJEKTNI MENADŽMENT KAO NEIZOSTAVNI DIO IT INDUSTRIJE 6

2.1. Razvoj projekta i njegovo definisanje 6

2.2. Pojava metodologija 8

3. METODE PROJEKTNOG MENADŽMENTA 9

3.1. Tradicionalni pristup 9

3.1.1. Vodopadni model 9

3.1.2. Spiralni model 11

3.2. Agilne metodologije 13

3.2.1. Ekstremno programiranje 15

3.2.1.1. Proces 16

3.2.1.2. Uloge i odgovornosti 17

3.2.1.3. Prakse 18

3.2.2. „Feature driven“ razvojni okvir 19

3.2.2.1. Proces 20

3.2.2.2. Uloge i odgovornosti 21

3.2.2.3. Prakse 22

3.2.3. Kanban 23

3.2.3.1. Proces 24

3.2.3.2. Uloge i odgovornosti *25*

3.2.3.3. Prakse *26*

3.2.4. „Crystal“ razvojni okvir 27

3.2.4.1. Proces *28*

3.2.4.2. Uloge i odgovornosti *29*

3.2.4.3. Prakse *30*

3.2.5. „Dynamic system“ razvojni okvir 32

3.2.5.1. Proces *32*

3.2.5.2. Uloge i odgovornosti *34*

3.2.5.3. Prakse *35*

4. SCRUM METODOLOGIJA 36

4.1. Scrum proces 37

4.2. Uloge i odgovornosti 38

4.2.1. Vlasnik proizvoda 39

4.2.2. Scrum master 40

4.2.3. Razvojni tim 40

4.3. Prakse 41

4.3.1. Scrum artefakti 41

4.3.2. Scrum događaji 45

5. KOMPARACIJA TRADICIONALNOG I AGILNOG PRISTUPA 49

6. ISTRAŽIVANJE I ANALIZA REZULTATA 54

6.1. Rezultati strukture uzorka 37

6.2. Rezultati odabira metodologije 38

6.3. Rezultati iz oblasti agilne metodologije 37

6.4. Rezultati iz oblasti tradiconalne metodologije 38

7. ZAKLJUČAK 64

REFERENCE 66

PRILOZI **71**

POPIS SLIKA

Slika 1. Faze vodopad modela 10
Slika 2. Spiralni model 12
Slika 3. Faze agilnog modela 15
Slika 4. Životni ciklus XP procesa 16
Slika 5. Procesi FDD-a 20
Slika 6. Kanban tabla 24
Slika 7. Dimenzije Crystal metodologije 28
Slika 8. Jedan Crystal Orange Inkrement 29
Slika 9. Dijagram DSDM procesa 33
Slika 10. Scrum Proces 38
Slika 11. Scrum tim 39
Slika 11. Backlog Proizvoda 42
Slika 13. Uređivanje Backlog Proizvoda 43
Slika 14. Sprint Backlog 43
Slika 15. Potencijalno isporučiv proizvodni inkrement 44
Slika 16. Izgled i vremensko razgraničenje Sprint-a 45
Slika 17. Sprint planiranje 46
Slika 18. Pregled Sprinta 47
Slika 19. Retrospektiva Sprinta 48
Slika 20. Poređenje tradicionalne i agilne metodologije 49

POPIS TABELA

Tabela 1. Prakse Ekstremnog programiranja 18
Tabela 2. Prakse DSDM-a 35

Tabela 3. Razlike između tradicionalne i agilne metodologije 50
Tabela 4. Razlike između agilne i tradicionalne metode u IT projektima..... 51

POPIS GRAFIKA

Graf 1. Godine postojanja kompanije..... 54
Graf 2. Broj uposlenika 55
Graf 3. Starosna dob uposlenika..... 55
Graf 4. Godine iskustva..... 56
Graf 5. Procenat primjene tradicionalne i agilne metodologije..... 56
Graf 6. Period praktikovanja agilnog načina rada 57
Graf 7. Agilni principi i primjena agilne metode 57
Graf 8. Najistaknutija stavka agilne metode..... 58
Graf 9. Prednosti agilne metodologije..... 58
Graf 10. Poboljšanja pri uvođenju agilne metode 59
Graf 11. Prepreke pri uvođenju agilne metode..... 59
Graf 12. Budući planovi primjene agilne metode 60
Graf 13. Stepen zastupljenosti agilnih metodologija..... 60
Graf 14. Razlozi primjene Scrum-a..... 61
Graf 15. Odabir tradicionalne metodologije..... 61
Graf 16. Razlozi korištenja tradicionalne metodologije..... 62
Graf 17. Nedostaci tradicionalne metodologije..... 62
Graf 18. Potencijalni prelazak na agilnu metodologiju 63
Graf 19. Plan prelaska na agilnu metodologiju 63

POPIS PRILOGA

Prilog 1. Lista kompanija 1
Prilog 2. Anketna pitanja..... 1

POPIS SKRAĆENICA

APM- Agilni projektni menadžment

BIH- Bosna i Hercegovina

DSDM- Dynamic system development method; Metoda razvoja dinamičkih sistema

FDD- Feature driven development; Razvoj baziran na karakteristikama

ICT- Informational communicational technology; Informaciono-komunikacione tehnologije

ISO- International Organization for Standardization; Međunarodna Organizacija za Standardizaciju

IT- Informacijske tehnologije

PBI- Product Backlog Items; Stavke proizvodnog backloga

PM- Projektni menadžment

PMBOK- Project Management Body of Knowledge; Zbirka znanja o upravljanju projektima

SDLC- Software Development Life Cycle; Životni ciklus razvoja softvera

WIP- Work in progress; Projekat u razvoju

XP- Extreme programming; Ekstremno programiranje

1. UVOD

Usljed konstantnog uvođenja novih tehnologija, softverskih aplikacija i rasta korištenja interneta, javlja se potreba širenja IT industrije i suočavanjem sa mnogim izazovima upravljanja IT projekatima. Razvoj softvera je jedna od glavnih aktivnosti koja se integrisala u sklopu IT kompanija, te je od velike važnosti imati dobro osmišljen proces kreiranja istog unutar dogovorenog vremena, unaprijed predviđenog budžeta, a da pri tome zadovoljava očekivanja kupca.

Kako navodi vodeći poslovni portal u Bosni i Hercegovini, Biznis Info (2020): „IT sektor u BiH je u usponu“. Tome su posvjedočili podacima koji pokazuju da je u 2017. godini u Bosni i Hercegovini registrovano čak 557 IT kompanija, a već 2019. godini taj broj se popeo na 701 registrovanu IT kompaniju. Broj radnika u IT sektoru je doživio značajan rast od 35% u posljednje tri godine. Sama IT industrija u Bosni i Hercegovini se snažno širi i time otvara prostor za veće rivalstvo IT kompanija na tržištu.

Po definiciji, upravljanje projektnom znači „Primjena znanja, vještina, alata i tehnika u projektnim aktivnostima radi zadovoljavanja projektnih zahtjeva“ (Project Management Institute, 2017, p.47). Projektni menadžment (PM) je opća praksa menadžmenta svake organizacije. Međutim, trenutna konkurencija na tržištu stvara pritisak na kompanije da brzo odgovore na promjene i kreiraju nove strategije unutar organizacija kako bi održale svoju konkurentnost, povećale profitnu maržu efikasnim korištenjem resursa i dosegle svoje ciljeve. U tom kontekstu ključni projekti su oni koji uključuju inovacije, koji zahtijevaju visoku fleksibilnost i brzinu odgovora zbog dinamičnog i promjenjivog okruženja.

U svakoj fazi životnog ciklusa IT projekta, veoma važnu ulogu igra projektni menadžer, projektni sponzor/kupac i projektni tim. Ono što projektnim menadžerima IT sektora predstavlja poseban izazov u odnosu na druge djelatnosti jeste to što pored primjene znanja, vještina i alata, moraju biti spremni na česte nadogradnje u sklopu svoje oblasti rada, u kojoj se pojavljuju nove tehnike i metode da bi se izveo određeni projekat na efikasniji način.

Kako je IT sektoru potreban projektni pristup, važno je poznavati različite metodologije upravljanja projektima, njihove specifičnosti, snage i slabosti, te odabir najprikladnije će rezultirati većoj uspješnosti istog.

Prema Project Management Institute (2017), metodologija je definirana kao sistem praksi, tehnika, postupaka i pravila koje koriste oni koji rade u projektoj disciplini. Kako je evoluirao svijet razvoja softvera i novih tehnologija, tako su se i razvijale metodologije.

Prvo je nastala tradicionalna metodologija sa poznatim „vodopad modelom“ koju karakterišu rigidno planiranje, strogo zapovijedanje i kontrola. Ta metoda stavlja veliki

naglasak na analizi i planiranje u ranim fazama projekta, donoseći procjene i pretpostavke vremena i troškova prije nego što su poznati svi faktori. Te se pretpostavke često temelje na modelima i šablonima, u kombinaciji s prethodnim iskustvom. Ovaj pristup se izvodi fazno i to kroz više faza koje se nadovezuju jedna na drugu. Kada se jedna faza završi prelazi se na sljedeću i tako redom. Ovu metodu čine sljedeće faze: prikupljanje zahtjeva, analiza, dizajniranje, implementacija, testiranje, integracija i održavanje.

U projektima razvoja softvera može se pojaviti problem da se u toku rada uvode neke promjene, koje zahtijevaju vraćanje na prethodnu fazu u kojoj se treba nešto izmijeniti, čemu tradicionalni pristup ne može odgovoriti. Zbog toga se ova metodologije smatra nedovoljno fleksibilnom za IT projekte, koji najčešće nemaju čvrsto definisanu strukturu zbog učestalih promjena u zahtjevima i specifikacijama, čime se mijenja i osnovna struktura projekta.

Prema istraživačkoj organizaciji *Forrester Research International*, jedan od najpopularnijih životnih ciklusa razvoja softvera sa značajnom razlikom danas je Agilni razvojni model (West i Grant, 2010). Agilno upravljanje projektima (APM) ili „agilne metode“ predstavljaju pristup upravljanju timom i okvir produktivnosti koji podržava kontinuirani i inkrementalni napredak u prioritetima u radu, čak i uslijed promjena.

Neke od agilnih metodologija o kojima ćemo nešto više reći u ovom radu su: *Scrum, Ekstremno programiranje (XP), „Feature driven“ razvojni okvir (FDD), Kanban, „Dynamic system“ razvojni okvir (DSDM) i „Crystal“ razvojni okvir*. One su zasnovane na prilagodljivosti bilo koje promjene kao sredstvo za povećanje šansi za uspjeh projekta (Cohen *et al*., 2004). Većina agilnih metoda pokušava minimizirati rizike tokom izvođenja projekta razvojem softvera u iteracijama, koje obično traju od jedne do četiri sedmice. Svaka iteracija je poput minijaturnog projekta koji je dio finalnog projekta i uključuje sve zadatke potrebne za implementaciju novih funkcionalnosti: planiranje, analiza zahtjeva, dizajn, kodiranje, ispitivanje i dokumentacija.

Projekt agilnog programiranja ima za cilj izdavanje novog softvera na kraju svake iteracije, a između svake iteracije tim preispituje svoje prioritete. APM je stekao popularnost posljednjih godina, prije svega u softverskoj industriji (Scrum Alliance, 2016), ali se progresivno probija u druge domene (Ciric *et al*.,2018).

Krajem 1990-ih timovi za razvoj softvera počeli su primjenjivati agilne metode za poboljšanje programskih procesa koje karakterišu prilagodljivost, lična i grupna autonomija, modularnost i samoorganizirana suradnja, kako je definirano u Agile manifestu (Beck *et al.*, 2001). Manifest je bio reakcija na slabosti i krutost popularnih metodologija izrade softvera zasnovanih na planovima, kao što je već navedeni tradicionalni model, koji je kritiziran uglavnom zbog nedostatka reakcije na promjene (Cockburn, 2002).

Ovaj rad će na osnovu istraživanje sprovedenog u IT kompanijama dokazati stepen primjene tradicionalnog i agilnog okvira, izdvojiti najrasprostranjeniju metodologiju unutar njih, te pokazati da li sami njihovi principi utiču na veću primjenjivost jednog okvira u odnosu na drugi.

1.1. Predmet istraživanja

Na osnovu prethodno obrazložene teme završnog rada, možemo zaključiti da svaka metodologija sama po sebi ima svoje principe, strukturu i način sprovođenja, gdje se ujedno vidi da među njima postoje razlike. Upravo te razlike su razlozi zbog čega će se određena metoda više primjenjivati u odnosu na drugu, jer njeno samo sprovođenje se zbog nečega ispostavilo mnogo uspješnije. Iz tog razloga, za svaku IT kompaniju veoma je važno da pronađe metod koji najviše odgovara njihovom načinu poslovanja i da time brzo odgovore na promjene, povećaju kvalitet proizvoda, produktivnost tima i zadrže kompetitivnost na tržištu.

U kontekstu navedenog definisan je problem istraživanja:

Problem istraživanja je nedovoljno istraženo pitanje metoda projektnog menadžmenta koje se primjenjuju u IT industriji u Bosni i Hercegovini.

U skladu sa ranije iznesenom problematikom i problemom istraživanja, definisan je i predmet istraživanja ovog rada:

Istražiti koje metodologije projektnog menadžmenta se trenutno primjenjuju u IT industriji u Bosni i Hercegovini, koja od njih se najviše koristi i zašto.

1.2. Istraživačko pitanje

Temeljom postavljenog predmeta i problema kao i predloženog okvira istraživanja, postavlja se sljedeće istraživačko pitanje:

- Koje metode projektnog menadžmenta se primjenjuju u IT industriji u BiH?
- Da li agilne metodologije imaju veću prednost u primjeniu odnosu na tradicionalnu zbog fleksibilnosti mijenjanja zahtjeva tokom razvoja projekta; brže isporuke softvera; veće kolaboracije sa klijentom i većom transparentnošću među članovima tima?
- Da li je metodologija SCRUM najčešće korištena metoda projektnog menadžmenta u IT industriji u BiH?

1.3. Cilj istraživanja

U skladu sa definisanim problemom i predmetom istraživanja, proizilaze i osnovni ciljevi:

1. Dati sistematiziran pregled metodogija koje se koriste za IT projekte;
2. Istražiti koje se sve metodologije projektnog menadžmenta koriste u IT kompanijama u Bosni i Hercegovini i u kojem procentu;
3. Napraviti komparaciju tradicionalnih i agilnih metodologija;
4. Dokazati prednosti agilne u odnosu na tradicionalnu metodologiju;
5. Istražiti procenat primjene agilne metodologije SCRUM za razvoj softverskih projekata;
6. Istražiti razloge zbog kojih se SCRUM primjenjuje ili ne primjenjuje u IT kompanijama;
7. Donijeti zaključke na osnovu rezultata istraživanja koji će implicirati koje metodologije se koriste u IT kompanija u Bosni i Hercegovini, koja od njih je najrasprostranjenija, da li je to SCRUM i zbog čega.

Istraživački ciljevi predstavljeni na ovaj način su istraženi u teorijskom i empirijskom dijelu završnog rada.

1.4. Hipoteze istraživanja

Imajući u vidu predmet istraživanja završnog rada, kao i ciljeve istraživanja, osnovne hipoteze ovog istraživanja glase:

H1: Agilne metode projektnog menadžmenta se više primjenjuju u IT kompanijama u BiH u odnosu na tradicionalne.

H2: IT projekti ukazuju na veću potrebu primjene agilne metodologije u odnosu na tradicionalnu zbog fleksibilnosti mijenjanja zahtjeva tokom razvoja projekta; brže isporuke softvera; veće kolaboracije sa klijentom i veće transparentnosti među članovima tima.

H3: SCRUM metodologija se najviše koristi u IT projektnom menadžmentu u BiH u odnosu na sve ostale agilne metodologije.

1.5. Metodologija

Metode istraživanja za ovu tezu su kombinacije teoretskih i praktičnih istraživanja iz oblasti IT projektnog menadžmenta. Dostupna literatura, članci, izvedene studije, publikacije i ostali relevantni izvori su uokvirili teoretski dio ove tematike.

U skladu sa specifičnostima naučno-istraživačkog rada, koristili su se primarni i sekundarni izvori podataka, te se distribuirala online anketa IT kompanijama koje posluju u Bosni i Hercegovini. Podaci su analizirani različitim istraživačkim metodama. U cilju provjere hipoteze i ostalih cjelina rada, koristile su se sljedeće metode:

- **Metoda analize** poslužila je za raščlanjivanje složenih pojmova i zaključaka na jednostavnije dijelove i elemente.
- **Metoda indukcije** trebala bi osigurati da se na temelju analize pojedinačnih činjenica dolazi do zaključka o općem sudu kako bi se obezbijedila relevantna teorija i praksa iz oblasti metoda upravljanja IT projektima.
- **Komparativna metoda** je primijenjena za upoređivanje tradicionale i agilne metodologije u IT projektima.
- **Deskriptivna metoda** će biti korištena kako bi se definisali pojmovi vezani za ovu temu iz oblasti IT projektnog menadžmenta.
- **Grafičko i tabelarno predstavljanje podataka** će se manifestovati u vidu prezentacije dobijenih rezultata nakon istraživanja sprovedenog u IT kompanijama.

Pored navedenih naučnih metoda, primijenjeni će se još i metoda **sinteze** i **dedukcije**.

1.6. Struktura rada

Okvirna struktura magistarskog rada se sastoji od sedam cjelina ili poglavlja.

U uvodu će se obrazložiti tema sa opisom problema i predmeta istraživanja. Navest će se ciljevi i hipoteza istraživanja, naučna metodologija, te pojašnjenje strukture samoga rada.

U drugom poglavlju, pojam projektnog menadžmenta bit će definisan i reći će se nešto više o bitnosti upravljanja IT projektima, projektnom timu i životnom ciklusu projekta.

Treći dio, metode projektnog menadžmenta za IT projekte bit će objašnjenje kroz prizmu dvije glavne podjele, a to su standardna ili tradicionalna metodologija i agilna metodologija, sa njihovim daljnjim podjelama.

Četvrto poglavlje nosi naziv Scrum metodologija, te će ono biti namijenjeno detaljnom obrazloženju već navedene metodologije kroz njene vrijednosti, principe i proces koji se ujedno sastoje od Scrum tima, artefakata i događaja.

U petom poglavlju, komparacija tradicionalnog i agilnog pristupa bit će prikazana, čime će se uočiti prednosti i nedostaci metodologija.

U šestom poglavlju, istraživanje metoda projektnog menadžmenta u IT industriji u BiH bit će prezentovano, te će se u okviru ovog istraživanja prezentovati rezultati za koje će se koristiti adekvatne metode obrade podataka.

U zaključku će se sistematski i koncizno formulisati najvažniji rezultati istraživanja, predstaviti njihovu potvrdu ili negaciju hipoteza, te time rezimirati zaključak koji će dati konkretan uvid primjene metode projektnog menadžmenta u IT industriji u Bosni i Hercegovini.

2. PROJEKTNI MENADŽMENT KAO NEIZOSTAVNI DIO IT INDUSTRIJE

2.1. Razvoj projekta i njegovo definisanje

U savremenom svijetu, nemoguće je zamisliti uspješno poslovanje bez korištenja informaciono-komunikacionih tehnologija (ICT). Različite naučne discipline ovise o njima, jer je neostvarivo završiti određenu aktivnost ili posao bez primjene istih. Da bi se stvorio jedan takav proizvod, potrebno je pokrenuti projekat, te upravljati istim do ispunjenja cilja.

Institut za upravljanje projektima osnovan je 1969. godine i stvorio je skup znanja za upravljanje projektima. 1996. godine objavili su prvu verziju „Zbirke znanja o upravljanju projektima“ (Project Management Body of Knowledge- PMBOK). Čak i u prvom izdanju PMBOK -a postoji 9 područja znanja i 37 procesa, a 9 procesa je povezano s drugom vrstom planiranja, što čini gotovo 25% svih procesa koji su definirani. S vremenom nastavljaju ažurirati ovo tijelo zasnovano na znanju i najnovije izdanje objavljeno je 2017. godine, koje sadrži 49 procesa, a 11 se od njih odnosi na planiranje (Project Management Institute, 2017).

Prema *Međunarodnoj Organizaciji za Standardizaciju* (ISO 8402) projekat je definisan kao jedinstven proces koji je sačinjen od skupa povezanih, složenih i kontrolisanih aktivnosti sa početnim i završnim datumima, pokrenut da bi se postigao određeni cilj uz poštivanje specifičnih zahtjeva kao što su ograničenje vremena, troškova i resursa.

Povezanost aktivnosti podrazumijeva postojanje logičkog ili tehničkog odnosa među parovima aktivnosti koje se moraju dovršiti određenim redoslijedom ili sekvencom. U projektu je vrijeme specificirano kroz određivanje datuma početka i završetka od strane uprave, eksterno od klijenta ili ovisno o primjeni određene metodologije upravljanja projektom. Budžet potreban za realizaciju jednog projekta se uglavnom određuje na samom početku. Resurse predstavljaju ljudi, oprema, fizički objekti ili inventar koji imaju ograničene mogućnosti, mogu se zakazati ili se mogu iznajmiti od vanjske strane. Neki su fiksirani, drugi su promjenjivi samo dugoročno. U svakom slučaju, oni su ključni za zakazivanje projektnih aktivnosti i uredan završetak projekta. Za projekte razvoja sistema ljudi su glavni resurs (Wysocki, 2014). Projekat je privremen i on sadrži primarnog kupca ili sponzora koji obično diktira pravac i projektni budžet. Uz projekat dolaze i određeni rizici i nesigurnosti koji mogu biti unutrašnji ili vanjski.

Kada je riječ o IT projektima, važno je istaknuti da se oni po određenim osobinama razlikuju od drugih vrsta projekata. U praksi se uglavnom misli na razvoj i uvođenje različitih softverskih rešenja i projekte uvođenja informacionih sistema.

U osnovi, IT kompanije se dijele na one koje se fokusiraju na razvoj hardvera i one koje razvijaju softver. Kada je riječ o Bosni i Hercegovini, većina kompanija su bazirane na razvoj softverskih rješenja, a hardver komponente se distribuišu, zbog nedovoljno razvijenog tog segmenta.

Samo značenje riječi "softver" je definisano kao "intelektualna kreacija koja obuhvata programe, pravila, procedure i odgovarajuću dokumentaciju koja prezentuje rad nekog sistema za obradu podataka" prema (ISO 2382).

Iako ovakvi projekti dijele mnoge sličnosti sa drugim vrstama projekata koji nisu IT orijentisani, postoje i određene razlike među njima. Neke od njih su: brza zastarjelost IT projekata zbog brzog trenda razvoja tehnologija; specifičnost projektnih timova kroz upravljanja ljudskih resursa (na njih često nije lahko primijeniti klasične metode i pristupe upravljanja); naglašenost grupnog rada; proizvod koji stvaraju je neopipljiv; sam ulaz u projekat jeste zamisao i opis karakteristika koje nisu fizička tvorevina (jedan od razloga otežanog mjerenja i predviđanja rezultata). Također, karakterišu ga i kompleksnost, ali i fleksibilnost zbog samog odgovora na promjene i mogućnost na brze adaptacije. S tim da su inovacije konstantno tražene, novi projekat postaje u jednu ruku otežanje i izazov programerima, jer staro iskustvo ne može biti dovoljno bez stvaranja novog znanja i kreiranja novih ideja (Prašo *et al*., 2016).

Prema portalu IT karijera (2019), inače podržanog od strane *Bit Alliance, Švedske agencije za razvog i saradnju SDC i Market Makers*, softverska industrija u Bosni i Hercegovini je već do sada uspjela postići istaknuto učešće u ukupnom bruto domaćem proizvodu, te je u prednosti u odnosnu na mnoge grane industrije. Ono po čemu je posebno prepoznatljiva jeste po kvalitetnoj ponudi radnih mjesta, visokim platama, razvoju „ekološki čistih proizvoda", upošljava i do 70% mladog kadra do 35 godina, ne uvozi značajno, pokazuje najbrži rast izvoza, prihoda i zaposlenih, ne iziskuje velike početne investicije za osnivanje, razvoj i pokretanje novih radnih mjesta, te mnoge druge grane industrije vrlo često ovise od softverskih rješenja koja oni kreiraju.

Također, navedeni portal obavještava da informacije prikupljene 2018. godine kroz istraživanje od strane konsultantske kuće *PwC* pokazuju da najzastupljenije industrijske grane kojima IT firme obavljaju posao su: E-commerce i web (10%), proizvodnja (9%), javna uprava (9%), edukacija (9%), bankarstvo i finansijske usluge (8%), maloprodaja (7%), transport (7%), zdravstvo (7%).

Često se govori o neuspjehu projekata koji je produkt mnogih faktora kao što su: nepotpuno definisanje projektnih ciljeva; loše planiranje i procjene; izostanak ili neadekvatna projektna metodologija, nerazumijevanje zahtijeva i jezika klijenata, visoka očekivanja klijenata i drugi (Prašo *et al*., 2016). Ovdje ćemo akcenat staviti na izostanku ili primjeni neadekvatne metodologije kroz uvođenje u samu terminologiju ovog pojma, te objašnjenje o postojećim metodologijama koje su zastupljene u IT projektima.

2.2. Pojava metodologija

Termin “metodologija” se prvi put upotrebljava 1956. godine od strane Herbert D. Beningtona na prezentaciji razvoja automatizovanog sistema kontrole za uhođenje i presretanje neprijateljskih aviona tokom hladnog rata. Ova prezentacija je bila spomenuta na konferenciji o naprednim programerskim metodama za digitalne računare (Herenda, 2016).

Institut za upravljanje projektom (2008) definira metodologiju upravljanja projektom kao skup metoda, tehnika, procedura, pravila, predložaka i najboljih praksi koje se koriste na projektu. Ostale definicije se ne razlikuju značajno. Charvat (2003) definira metodologiju upravljanja projektom kao skup smjernica i principa koji se mogu skrojiti i primijeniti na specifičnu situaciju, gdje smjernice mogu biti jednostavne kao lista zadataka, ili to može biti specifičan pristup projektu s definiranim alatima i tehnikama. Slično, Gane (2001) definira metodologiju upravljanja projektom iz perspektive znanja kao skup znanja o zadacima, tehnikama, isporukama, ulogama i alatima koji se koriste tokom projekta uparen sa znanjem o prilagođavanju svega toga određenom projektu. Najopsežniju definiciju metodologije upravljanja projektima daje Cockburn (2003). On je definira kao bilo koji princip na koji se tim za upravljanje projektom oslanja kako bi uspješno postigao rezultate projekta.

Cockburn (2006) navodi nekoliko metodoloških svrha, poput uvođenja novih članova tima u proces, lakše zamjene članova tima, jasne odgovornosti, utisak korisnika, vidljiv napredak i izvještavanje o statusu i obrazovanje. Dobra metodologija upravljanja projektima vodit će voditelja projekta kroz kontrolirani, upravljani i vidljivi skup aktivnosti u cilju postizanja rezultata projekta (Ured za trgovinu Vlade, 2009). Karakteristike dobre metodologije prema Kerzneru (2001) su: preporučeni nivo detalja, upotreba predložaka (engl. templates), standardizirano planiranje, upravljanje vremenom i tehnikama kontrole troškova, standardizirano izvještavanje, fleksibilnost za korištenje na svim projektima, fleksibilnost za brzi razvoj, da je razumljiva za korisnika, prihvaćena i upotrebljiva unutar organizacije, da koristi standardizirane faze životnog ciklusa projekta, te se temelji na smjernicama i dobroj poslovnoj etici.

Kako su tehnologije napredovale, tako su se metodologije primjenjivale, zastarijevale i stvarale se nove. U ovoj borbi ostajale su one koje su prepoznate kao dovoljno kvalitetne i prilagodljive da odgovore na zahtjeve brzo mijenjajućeg tržišta.

Metodologije sa kojim ćemo se baviti u ovom radu su tradicionalne i agilne. Sve je počelo od pojave vodopad modela koji pripada tradicionalnom pristupu. Potom se pojavljuje iterativni model kao rafinirana verzija vodopad modela koji se danas svejedno prisvaja tradicionalnom pristupu, a njegov predstavnik je spiralni model. Onda nastupaju agilne metode sa inkrementalnim i iterativnim pristupom i sa nizom već spomenutih metodologija.

3. METODE PROJEKTNOG MENADŽMENTA

3.1. Tradicionalni pristup

Samo postojanje i široka primjena tradicionalnog pristupa upravljanja projektom poznato je još od prve verzije „Zbirke znanja“ razvijene 1980-ih, kada je navedeni pristup zapravo bio jedina praksa. Tokom sljedećih izdanja, primijenjena su ažuriranja stvarnih praksi, ali brzina promjena nije uvijek bila usklađena s očekivanjima praktičara (Project Management Institute, 2017).

Osnovna ideja tradicionalnog, racionalnog i normativnog pristupa je da su projekti relativno jednostavni, predvidljivi i linearni s jasno definiranim granicama što olakšava detaljno planiranje i praćenje plana bez većih promjena. Ovo je takozvani redukcionistički pristup, jer se razbijaju cjeline na manje dijelove do stepena kojeg je moguće opisati matematičkom preciznošću (Wysocki, 2014). Krajnji cilj tradicionalnog pristupa upravljanju projektom je optimizacija i efikasnost u slijeđenju početnog detaljnog plana projekta, ili, na uobičajen način, da se projekat završi u planiranom roku, budžetu i opsegu.

Špundak (2014) ističe da ga karakterišu robusnost, primjena istih metoda i tehnika na sve projekte, hijerarhijski i linearan odnos zadataka, izoliranost od okoline, otpor prema promjenama, neprilagodljivost dinamičnom okruženju, neizvijesnost u izhodu složenih i inovativnih projekata itd.

Prema navodima Schwaber-a (2007), u tipičnim projektima primjenom tradicionalne metodologije oko 50% vremena se troši na zahtjeve, specifikacije i arhitekturu, te da se sve to radi čak i prije kreiranja bilo koje funkcionalnosti. Međutim, 35% zahtjeva se mijenja, a 65% svojstava opisanih u zahtjevima nikada se ili rijetko koriste. Sav taj fokus kreiranja struktuiranih procesa sa naglašenim detaljnim planiranjem mogu doprinjeti potencijalnoj neefikasnosti primjene ove metode u složenom, dinamičnom u inovativnom projektu.

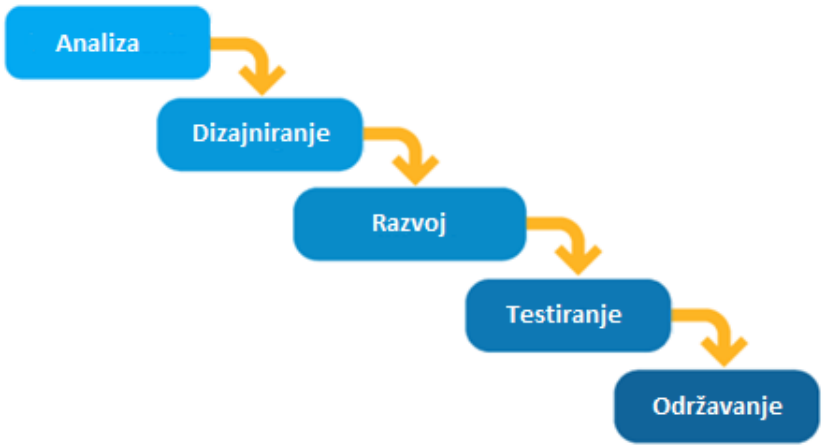
U nastavku pročitajte nešto detaljnije o tipovima tradicionalnog pristupa koji nose naziv *Vodopadni i spiralni model.*

3.1.1. Vodopadni model

Vodopadni model (engl. Waterfall model) za upravljanje projektima prvi put je predstavljen 1970. godine od strane Winston W. Royce (Hillaire, 2018). Royce u svom članku po nazivu ‘*Upravljanje razvojem velikih softverskih sistema*’ prezentovao je nekoliko projektnih modela, uključujući ono što mi danas poznajemo kao vodopad metoda. Sam termin „Vodopadni model“ nije bio upotrebljen u ovom članku, te se njegova prva upotreba pojavljuje 1976. godine od strane Bell i Thayer (Herenda, 2016).

Bassil (2012, p.2) definira vodopadni model kao „proces uzastopnog razvoja softvera u kojem se napredak smatra sve težim prema dolje (slično vodopadu) kroz popis faza koje se moraju izvršiti da bi se uspješno izgradio računarski softver“. On je jedan od najstarijih modela najpogodniji za rješavanje potpuno jasnih zahtjeva bez naknadne izmjena istih**.** Kao model je jednostavan, razumljiv korisniku, te je njegov razvojnog procesa sačinjen od niza faza.

Slika 1 Faze vodopad modela



Izvor: Project Management (2020)

U nastavku su demonstrirane faze waterfall modela prema Bassil (2012):

(a) **Faza analize** – Najvažnija faza u kojoj su definirani funkcionalni zahtjevi do detalja. Dokument koji sadrži ove zahtjeve mora biti precizan i nedvosmislen, potvrđen od strane klijenta, jer se na njemu temelji nastavak projekta.

(b) **Faza dizajniranja –** Na osnovnu zahtijeva dizajniraju se komponente kao što su arhitektura softvera i struktura podataka. Ovdje se dizajn softvera i hardvera posmatra odvojeno. Pravi se prezentacija samog softvera da bi se procijenio kvalitet istog prije početka kodiranja. Također se sprema testna knjiga kroz koju će se provjeriti proizvod u testnoj fazi.

(c) **Faza implementacije** – Nakon specifikacije zahtijeva i dizajna, vrši se kodiranje gdje se dizajn konvertuje u konkretniju formu. U ovoj fazi je upotrebljiv alat realiziran.

(d) **Fazu testiranja** – Ovdje dolazi do provjere i validacije cijelog sistema sa funkcionalnostima kako bi se provjerila njihova prisutnost u istom i njihove performanse. Testiranje je temeljeno po testnoj knjizi, te se nakon testiranja kreira izvještaj.

(e) **Fazu održavanja** – Završna faza u kojoj se proizvod distribuira klijentu radi povratne informacije. Ovdje se ispravljaju problemi i implementiraju poboljšanja. Trajanje ove faze biva sve dok sistem ne zastari ili ako se zamijeni nekim novim.

Jedna od prepoznatljivih odlika ovog modela jesu jasne granice među fazama sa njihovim izlaznim produktima. Svaka faza mora biti u potpunosti okončana prije kretanja naredne faze. Pri završetku svake faze vrši se analiza radi utvrđivanja je li projekat ide u pravom smjeru, da li se treba nastaviti sa istim ili ga obustaviti.

Kako Hillaire (2018) navodi, projekti vodopada dijele karakteristike kao što su strogo planiranje i kontrola u svim fazama životnog ciklusa projekta (PLC–a) postignute komandnim i kontrolnim stilom upravljanja, formalizirana dokumentacija, uključivanje korisnika i razvoj cijelog rješenja u jednom faznom pristupu koji se javlja nakon planiranja i prije testiranja.

Primarni alati i tehnike vodopada ili tradicionalne metode upravljanja projektima su struktura raščlanjivanja radova (work breakdown structure- WBS), koja daje osnovnu definiciju projektnog rada; matrica odgovornosti, koja članovima projektnog tima dodjeljuje poslove navedene u WBS-u; Gantov grafikon, koji pruža vizuelni prikaz rasporeda projekta; projektne mrežne tehnike, koje ilustruju raspoređivanje uticaja koje aktivnosti imaju jedna na drugu i pružaju sredstva za određivanje kritičnih aktivnosti i vremena prolaska; raspoređivanje troškova, koje identifikuje neophodna kapitalna ulaganja; i proces kontrole projekta, koji će otkriti prekoračenja projekata u rasporedu ili budžetu.

3.1.2. Spiralni model

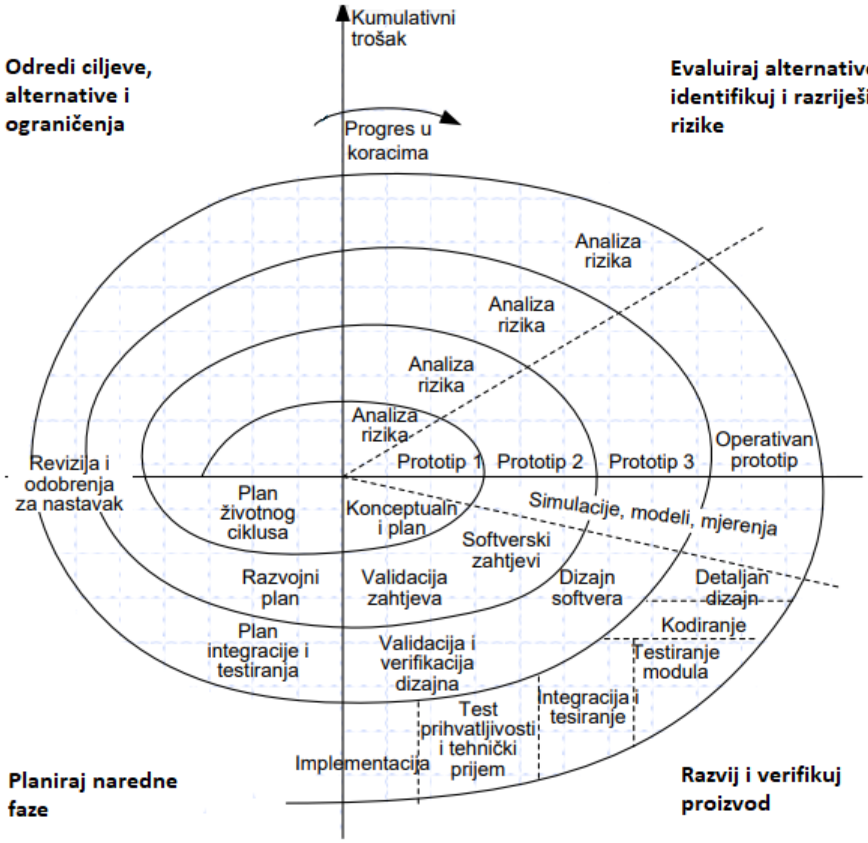
Spiralni model je proizašao iz različitih rafiniranja waterfall modela, te je sistem razvijen iz serije prototipova. Ovaj model je prvi put opisao Barry Boehm, 1986 godine. Navodeći sliku, Kumar (2020) pojašnjava da je ovaj model u svom dijagramskom prikazu predstavljen je kao spirala sa mnogo petlji (Slika 2). Precizan broj petlji i faza spirale je nepoznat, jer variraju od projekta do projekta. Svaka petlja naziva se faza procesa razvoja softvera. Kako voditelj projekta dinamički određuje broj faza, tako on ima bitnu ulogu u razvoju proizvoda koristeći ovaj model.

Radijus spirale u bilo kojoj tački označava troškove dosadašnjeg projekta, a ugaona dimenzija označava dosadašnji napredak u trenutnoj fazi. Napredak se ostvaruje kroz ponavljanje istog niza koraka, ali na različitim stepenima detaljnosti, počevši od osnovnih konceptualnih dokumenata, pa do programiranja zasebnih dijelova softvera. Svaki prototip dodaje poboljšanja prethodnim. Softverski projekat se kontinuirano kreće kroz faze, sličnim vodopadu, unutar svake spiralne iteracije (Boehm, 1988).

Svaka faza spiralnog modela podijeljena je u četiri kvadranta kao što se može vidjeti na slici ispod. Funkcije ova četiri kvadranta su prema Kumaru (2020) objašnjene u nastavku:

1. **Određivanje ciljeva i identifikacija alternativnih rješenja**: Zahtjevi se prikupljaju od kupaca, a ciljevi se identifikuju, razrađuju i analiziraju na početku svake faze. Tada se u ovom kvadrantu predlažu alternativna rješenja moguća za fazu.
2. **Identifikacija i rješavanje rizika:** Tokom drugog kvadranta, sva moguća rješenja se procjenjuju kako bi se odabralo najbolje od njih. Potom se identifikuju rizici povezani sa tim rješenjem i rizici se rješavaju primjenom najbolje moguće strategije. Na kraju ovog kvadranta, prototip je napravljen za najbolje moguće rješenje.
3. **Razvijanje sljedeće verzije proizvoda:** Tokom trećeg kvadranta, identifikovane karakteristike se razvijaju i verificiraju kroz testiranje. Na kraju ovog kvadranta, dostupna je sljedeća verzija softvera.
4. **Pregledavanje i planiranje za narednu fazu:** U četvrtom kvadrantu, kupci procjenjuju do sada razvijenu verziju softvera. Na kraju se kreće sa planiranjem sljedeće faze.

Slika 2 Spiralni model



Izvor: Boehm (1988)

Rizik je jedna od neželjenih situacija koja može uticati na uspješnost softverskog projekta. Najvažniji segment ovog modela je ispravno rukovođenje sa nepoznatim rizicima tokom projekta. Rješavanje rizika se lakše postiže razvojem prototipa u svakoj fazi razvoja softvera.

Model izrade prototipa (engl. Prototyping model) jedan je od onih koji podržava bavljenje rizicima, ali oni moraju biti u potpunosti identificirani prije početka rada na razvoju projekta (Kumar, 2020). U stvarnosti, rizik projekta se može pojaviti nakon početka razvojnog rada, i u tom slučaju ne možemo koristiti model prototipa.

Spiralni model nosi još i naziv *meta-model* jer obuhvata sve ostale SDLC modele. Za primjer jesto to da spirala jedne petlje zapravo predstavlja iterativni model vodopada. Ovaj model sadrži postupni pristup klasičnog vodopad modela, samo što za razliku od vodopada, on razvija prototip na početku svake faze kao tehniku upravljanja rizikom (Kumar, 2020). Uz to sve, spiralni model je jedan od vidova podrške evolucijskom modelu jer iteracije duž spirale mogu se smatrati evolucijskim nivoima pogodnim za gradnju kompletnog sistema.

Primjer za ovaj model je bi bio Microsoft Windows 3.1 operativni sistem, koji je bio pušten u javnost, te njegove sljedbenice koje su dolazile godinama poslije.

3.2. Agilne metodologije

Izraz agilno upravljanje projektima (APM) počeo se širiti 2001. godine, a kulminiralo je potpisivanjem Agilnog Manifesta, dokumenta koji su pripremili profesionalci i teoretičari tehnološko-informacijskog područja dovodeći u pitanje tradicionalni pristup upravljanja projektima, posebno ako se primjenjuju u projektima koji uključuju neizvjesnosti i podložnost promjenjivom poslovnom okruženju (Azanha *et al*., 2017). Sama riječ agilnost označava „spremnost za kretanje; okretnost, aktivnost, spretnost u pokretu" (Oxford English Dictionary, 1989). U ovom kontekstu može se shvatiti kao sposobnost stvaranja i reagiranja na promjene, kako bi se održala profitabilnost u turbulentnom poslovnom okruženju, ili sposobnost balansiranja fleksibilnosti i stabilnosti (Highsmith, 2004).

Što se tiče agilnog projektnog menadžmenta (APM), u literaturi postoje različite definicije. Po Highsmith-u (2004), APM je skup principa, vrijednosti i praksi koji pomažu timu da isporuči vrijednost proizvoda ili usluga u izazovnim projektima za okruženje. Po mišljenju Augustin-a (2005), APM je radni način da se energizira, osnaži i omogući projektnom timu brza, sigurna isporuka poslovne vrijednosti, kroz integraciju kupaca u kontinuirani proces učenja i prilagođavanja promjenama, u skladu sa njihovim potrebama i poslovnom okruženju. Među većinom autora postoji konsenzus o suštini APM -a kada kažu da je to pristup koji traži fleksibilnost, jednostavnost, ponavljanja u kratkim vremenskim periodima i postupno dodavanje vrijednosti (Azanha *et al*., 2017).

Primjena agilnih principa imala je veći poticaj od konferencije OOPSLA 1995.godine, na kojoj su Ken Schwaber i Jeff Sutherland izložili principe i raspravljali o njihovoj primjeni u razvoju softvera. Azanha *et al*. (2017) navodi da Schwaber, Sutherland, Cervone, Beck i drugi analizirajući tradicionalni proces razvoja softvera, shvatili su da se ova metodologija ne pridržava empirijskih, nepredvidivih i neponovljivih procesa.

Prema Beck *et al*. (2001) agilni manifest za razvoj softvera sadrži 12 principa od koji se ističu 4 ključna:

I. **Ljudi i interakcije** prije nego li procesi i alata.
II. **Softver koji radi** prije nego li sveobuhvatne dokumentacije.
III. **Saradnja sa klijentom** prije nego li pregovaranja o ugovoru.
IV. **Odgovoriti na promjene** prije nego li slijediti plan.

U prvom principu je istaknuta relacija i zajedništvo programera i ljudska uloga je naglašena u ugovorima, za razliku od razvojnih alata i institucionaliziranih procesa. To se manifestuje kroz bliske timske odnose, radna okruženja i druge postupke koji jačaju timski duh (Azanha *et al*., 2017).

U drugom principu naglašava se osnovni cilj softverskog tima, a to je proizvodnja testiranog radnog softvera (Azanha *et al*., 2017). Nova izdanja se isporučuju intervalima koji su česti, po nekim metodama je to čak i po satu ili dnevno, ali obično to bude jedan do 3 mjeseca. Programerov kod treba biti jasan, tehnički napredan i jednostavan, zbog čega bi se teret dokumentacije smanjio na odgovarajući nivo.

U trećem principu, komunikacija i saradnja između programera i klijenata su važniji od strogih ugovora, iako bitnost dobro urađenih ugovora raste istim tempom kao i veličina softverskog projekta. Samo pregovoranje i saradnja su sredstvo za postizanje kvalitetnog odnosa (Azanha *et al*., 2017). Kada se gleda sa poslovne strane, agilni razvoj stavlja svoj fokus odmah nakon početka projekta kroz isporuku poslovne vrijednosti, zbog čega se smanjuju rizici neispunjenja ugovora.

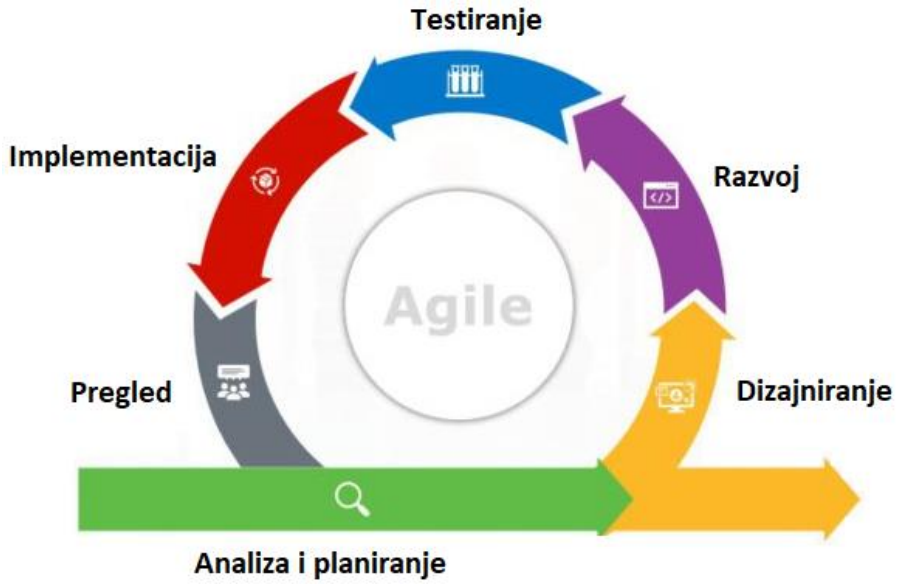
U četvrtom principu, programeri i predstavnici kupaca imaju dužnost biti dobro informisani, kompetentni i ovlašteni uzeti u obzir moguće potrebe prilagođavanja koje se dešavaju tokom životnog ciklusa razvojnog procesa. Ovdje je naglašena spremnost na promjene, te formiranje postojećih ugovora sa alatima koji podržavaju i omogućuju vršenje poboljšanja (Azanha *et al*., 2017).

Najistaknutije karakteristike agilnih metoda su inkrementalna priroda razvojnog proizvoda (mala izdanja softvera, sa brzim ciklusima), iteracije sa kratkim vremenskim intervalima, bliska saradnja projektnog tima i klijenta, jednostavan i prilagodljiv razvoj softvera (sama metoda je laka za učenje i modifikovanje, dobro dokumentovana, te otvara prostor za promjene u posljednjem trenutku (Hillaire, 2018).

Većina agilnih metoda pokušava smanjiti rizik tokom izvođenja projekta razvijanjem softvera u iteracijama, koje obično traju od jedne do četiri sedmice. Svaka iteracija je poput minijaturnog završnog projekta i uključuje sve zadatke potrebne za implementaciju novih funkcionalnosti: analiza zahtjeva i planiranje, dizajniranje, kodiranje, testiranje, implementacija i pregled (Slika 3) (Hidalgo, 2019). Inkrementalna isporuka podrazumijeva

u više navrata isporuku funkcionalnih dijelova sistema (ili verzija kompletnog sistema) koji se puštaju u eksploataciju. Agilni programski projekt ima za cilj objavljivanje novog softvera na kraju svake iteracije, a između svake iteracije tim preispituje svoje prioritete (Hidalgo, 2019).

Slika 3 Faze agilnog modela



Izvor: SlideUpLift (2021)

Agilna metoda sadrži spektar metodologija koje će biti detaljnije objašnjene u nastavku rada, a one su: *Scrum, Ekstremno programiranje (XP), „Feature driven" razvojni okvir (FDD), Kanban, „Dynamic system" razvojni okvir (DSDM) i „Crystal" razvojni okvir*.

Hawrysh i Ruprecht (2000) navode da jedna metodologija ne može funkcionirati za cijeli spektar različitih projekata, već bi umjesto toga menadžment projekta trebao identificirati specifičnu prirodu projekta i zatim odabrati najbolju primjenjivu razvojnu metodologiju.

3.2.1. Ekstremno programiranje

Ekstremno programiranje (XP) je agilni okvir za razvoj softvera koji ima za cilj proizvodnju softvera višeg kvaliteta. XP je najspecifičniji od agilnih metodologija u pogledu odgovarajuće inženjerske prakse za razvoj softvera, gdje je fokus na tehničkom aspektu (Agile Alliance, 2021).

XP je prvi put primijenjen u programu *Chrysler Comprehensive Compensation (C3)* koji je pokrenut sredinom 90-ih, dolaskom softverskog inžinjera Kent Beck-a koji je uključen u projekat radi poboljšanja performansi sistema. Dolaskom još nekoliko ljudi u tim, uključujući Rona Jeffriesa, promijenio se način na koji je tim pristupio razvoju u odnosu na prije. Ovaj projekat doprinio je popularnosti ove metodologije (Agile Alliance, 2021).

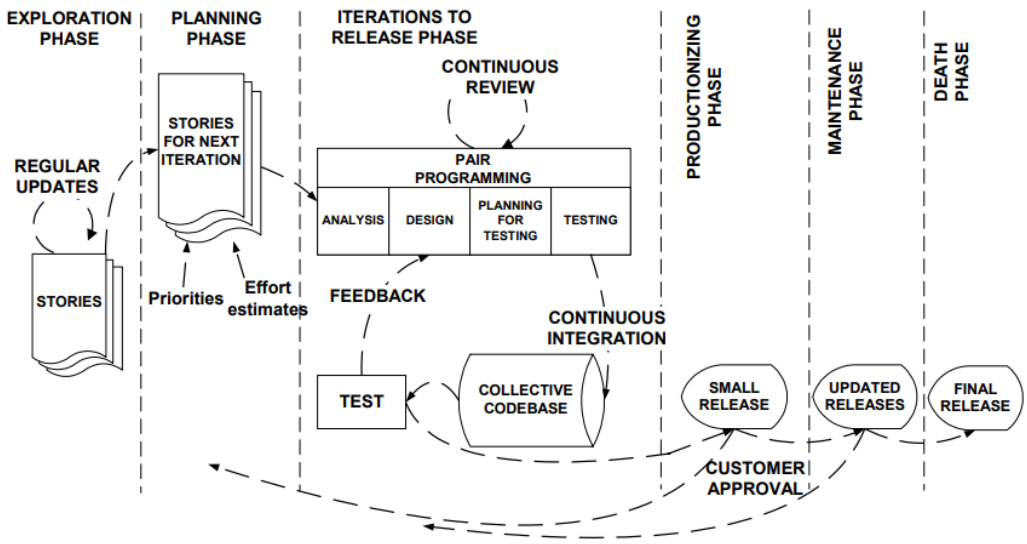
Ekstremno programiranje nastalo je iz problema uzrokovanih dugim razvojnim ciklusima tradicionalnih modela (Abrahamsson *et al*., 2002). Evoluirao je od jednostavnih napora da se uradi posao programiranja, ali skraćivanjem tradicionalnog načina razvoja softvera. Pojam „ekstremno“ proizlazi od podizanja zdravorazumskih principa i praksi (objašnjenih u nastavku) na ekstremne nivoe (Beck, 2000). Softversko rješenje se kreira u seriji malih, potpuno integrisanih izdanja koja su prošla sve testove definisanih od strane klijenta. Povratne informacije se dobijaju testiranjem softvera počev od prvog dana. Oni isporučuju sistem kupcima što je ranije moguće i primjenjuju promjene koje su tražene (Wells, 2013).

Pet osnovnih vrijednosti ove metodologije su: *komunikacija, jednostavnost, povratne informacije, poštovanje i hrabrost*.

3.2.1.1. Process

Životni ciklus XP -a obuhvata pet faza: istraživanje, planiranje, od iteracije do isporuke, proizvodnja, održavanje i smrt (Abrahamsson *et al*., 2002).

Slika 4 Životni ciklus XP procesa



Izvor: Abrahamsson et al. (2002)

Prva faza jeste faza **istraživanja** u kojoj klijenti ispisuju kartice sa karakteristikama koje žele imati u prvoj verziji, te se one dodaju u program. Uporedo se projektni tim upoznaje sa alatima, tehnologijom i praksama koje će biti korištene. Korištena tehnologija će se testirati, a rad arhitekture sistema će se istražiti putem izgradnje prototipa sistema. Trajanje faze je između nekoliko sedmica i nekoliko mjeseci zavisno o poznavanju tehnologije od strane programera.

Poslije istraživanja dolazi **faza planiranja** u kojoj se pravi prioritetni redoslijed karakteristika i sklapa se dogovor o sadržaju prve male verzije. Prvo se procjenjuje koliko

je truda potrebno za svaku aktivnost koji će programer uraditi, te se pravi dogovor o rasporedu istih. Trajanje faze je nekoliko dana.

Faza **iteracija do isporuke** podrazumijeva nekoliko iteracija sistema prije prvog izdanja. Redoslijed stavki postavljen u fazi planiranja podijeljen je na niz iteracija kojima će svaka trebati jednu do četiri sedmice za implementaciju. Početna iteracija sadrži sistem s arhitekturom cijelog sistema. Da bi se došlo do toga, odabiru se karakteristike koje će biti dio izgradnje strukture cijeloga sistema. Klijent donosi odluke o izboru karakteristika za svaku iteraciju. Posljednjom iteracijom sistem je spreman za proizvodnju.

Faza **proizvodnje** uključuje dodatno testiranje i provjeru performansi sistema prije nego bude prodan. U ovoj se fazi se mogu napraviti nove promjene, ali se mora prvo razmotriti o potrebi istih. Odgođene ideje i prijedlozi dokumentiraju za fazu održavanja.

Nakon proizvodnje prve verzije, XP projekt mora održavati sistem u pokretu, a uporedo proizvoditi nove iteracije. Do postizanja istog, **faza održavanja** iziskuje napor za zadatke korisničke podrške. Iz tog razloga može se usporiti razvojna brzina nakon puštanja sistema u proizvodnji.

Faza **smrti** nastupa kada nema više stavki za implementaciju od strane klijenta. Prema ovome bi sistem trebao biti spreman da zadovolji potrebe kupaca. Ovo je faza XP-a procesa kada se završava potrebna dokumentacija sistema, jer se više ne mijenjaju arhitektura, dizajn ili kod. Do smrti može doći i u slučaju da sistem ne daje željene rezultate ili ako postane preskup za daljnji razvoj.

3.2.1.2. Uloge i odgovornosti

XP djeluje tako što okuplja čitav tim u prisustvu jednostavnih praksi, s dovoljno povratnih informacija da timu omoguće da vidi gdje se nalazi i prilagodi prakse svojoj jedinstvenoj situaciji (Jeffries, 2011). Timski rad je ovdje naglašen (Wells, 2013), ali postoje postoje različite uloge, te u skladu sa tim i različiti zadaci. U nastavku su ove uloge predstavljene prema Beck-u (2000):

- **Programeri** - Pišu testove i održavaju programski kod što jednostavnijim i nedvosmislenim.
- **Kupac** - Piše priče i funkcionalne testove, te odlučuje kada je svaki zahtjev zadovoljen.
- **Ispitivač (Tester)** - Pomaže korisniku da napiše funkcionalne testove. Redovno testira funkcionalnosti, prezentuje rezultate testova i održava alate za testiranje.
- **Tragač (Tracker)** - Daje povratne informacije. On prati procjene koje je napravio tim (npr. procjene napora) i daje povratne informacije o tome koliko su one ispravne kako bi se poboljšale buduće procjene. Također, on prati napredak svake iteracije, dostižnost cilja unutar danih resursnih i vremenskih ograničenja i da li su u procesu potrebne neke izmjene.

- **Trener -** Odgovoran za proces u cjelini. Dobro razumijevanje XP -a važno je u ovoj ulozi koja omogućava treneru da vodi ostale članove tima kroz metodologiju.
- **Konsultant -** Vanjski član koji posjeduje potrebno tehničko znanje. On vodi tim u rješavanju njihovih specifičnih problema.
- **Menadžer (Big Boss) -** Donosi odluke. On komunicira s projektnim timom kako bi procijenio trenutnu situaciju i prepoznao potencijalne poteškoće ili nedostatke u procesu.

3.2.1.3. Prakse

Suština XP-a je međusobno povezan niz dolje navedenih praksi razvoja softvera. Iako je moguće ove prakse raditi odvojeno, mnogi timovi su otkrili da neke prakse pojačavaju druge i da ih treba raditi zajedno kako bi se u potpunosti izbjegli rizici s kojima se često susreće u razvoju softvera (Agile Alliance, 2021).

Tabela 1 Prakse Ekstremnog programiranja

• **Planiranje igre** Podrazumijeva blisku interakcija između klijenta i programera. Programeri procjenjuju napor potreban za implementaciju karakteristika, a potom kupac odlučuje o opsegu i vremenu objavljivanja.	• **Mala izdanja** Tim izdaje pokrenut, testirani softver, pružajući poslovnu vrijednost koju je odabrao kupac, svaku iteraciju. Kupac može koristiti ovaj softver u bilo koju svrhu, bilo da se radi o procjeni ili čak objavljivanju krajnjim korisnicima.
• **Metafora** Timovi za ekstremno programiranje razvijaju zajedničku viziju kako program funkcionira, a koju nazivamo „metaforom“. U najboljem slučaju, metafora je jednostavan evokativni opis rada programa kako bi bili sigurni da svi razumiju kako sistem funkcionira.	• **Jednostavan dizajn** XP timovi grade softver jednostavnog, ali uvijek adekvatnog dizajna. Počinju jednostavno, a kroz testiranje programera i poboljšanje dizajna to zadržavaju. XP tim održava dizajn tačno prilagođenim trenutnoj funkcionalnosti sistema.
• **Testiranje** Kao dio predstavljanja svake željene značajke, kupac XP definira jedan ili više automatiziranih testova prihvaćenosti kako bi uvidio da značajka radi. Tim izrađuje ove testove i koristi ih kako bi dokazao sebi i kupcu da je funkcija pravilno implementirana.	• **Poboljšanje dizajna** Proces refaktoriranja fokusira se na uklanjanje duplikata (siguran znak lošeg dizajna) i na povećanje „kohezije“ koda, dok istovremeno smanjuje „spregu“. Da bi se postigla visoka poslovna vrijednost projekta, softver mora biti dobro dizajniran procesom kontinuiranog poboljšanja.
• **Programiranje u paru**	• **Kolektivna odgovornost za kod**

Dvije osobe pišu kod na jednom računaru. Ova praksa osigurava da sav proizvodni kod pregleda najmanje jedan drugi programer i rezultira boljim dizajnom, boljim testiranjem i boljim kodom.	Svatko može promijeniti bilo koji dio koda u bilo koje vrijeme. To znači da sav kod dobiva pažnju mnogih ljudi, što povećava kvalitetu koda i smanjuje nedostatke.
• **Kontinuirana integracija** Novi dio koda integrira se u bazu kodova čim bude spreman. Dakle, sistem se integrira i gradi mnogo puta dnevno. Svi testovi se izvode i moraju biti položeni da bi se prihvatile promjene u kodu.	• **Kupac na licu mjesta** Dužnost klijenta jeste da bude prisutan i dostupan svo vrijeme za tim.
• **Standard kodiranja** XP timovi slijede zajednički standard kodiranja, tako da sav kod u sistemu izgleda kao da ga je napisao jedan - vrlo kompetentan - pojedinac. Specifičnosti standarda nisu važne: važno je da sav kod izgleda poznat kao podrška kolektivnom vlasništvu.	• **Samo pravila** Tim ima svoja pravila koja slijedi, ali se ona također mogu promijeniti u bilo kojem momentu. U slučaju promjena, mora se težiti ka dogovoru oko istih i procijeniti njihov uticaj.

Izvor: Abrahamsson et al., (2002) i Jeffries (2011)

3.2.2. „Feature driven“ razvojni okvir

„Feature driven“ razvojni okvir (FDD) ili u punom prijevodu „Razvoj baziran na karakteristikama“ je agilna i prilagodljiva metoda razvoja sistema. Ovaj pristup ne pokriva cijeli proces razvoja softvera, već se fokusira na faze projektiranja i izgradnje i ne zahtijeva upotrebu nekog posebnog modela procesa (Palmer i Felsing, 2002). FDD je zasnovan na iterativnom razvoju sa najboljim praksama za koje je utvrđeno da su učinkovite u industriji. Potencira na kvaliteti u cijelom procesu i sadrži česte i opipljive isporuke, uz precizno praćenje napretka projekta (Abrahamsson *et al*., 2002).

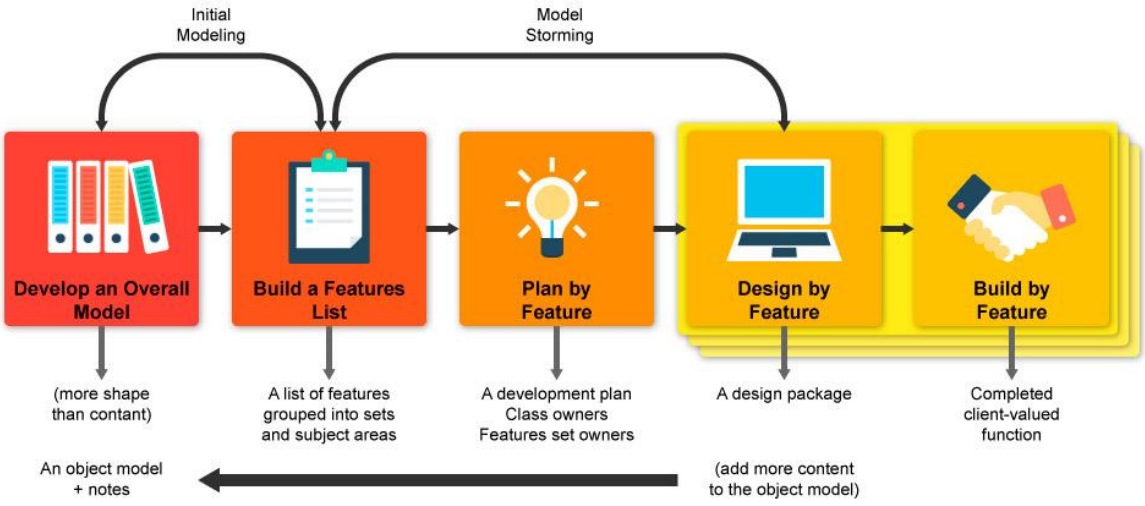
FDD je sadržan od pet uzastopnih procesa i uključuje metode, tehnike i smjernice potrebne članovima projekta za isporuku sistema. Također, sadrži uloge, artefakte, ciljeve i vremenske rokove potrebne u projektu. U poređenju sa drugim agilnim metodologijama, FDD je prikladan za razvoj kritičnih sistema (Palmer i Felsing 2002).

FDD je prvi put predstavljen u 1999. godini (Coad *et al*. 2000). Potom je veću nadogradnju dobio od strane Jeff-a Luca, Stephen-a Palmer i Peter-a Coadi.

3.2.2.1. Proces

FDD metodologija sadrži pet uzastopnih procesa tokom kojih se vrši projektiranje i izgradnja sistema (Slika 5). Sistem se razvije na agilni način kroz iteracije s brzim prilagođavanjem kasnim promjenama zahtjeva i poslovnih potreba. Period rada tima za iteraciju je od jedne do tri sedmice (Abrahamsson *et al*., 2002).

Slika 5 Procesi FDD-a



Izvor: Team (2020)

U nastavku je svaki od pet procesa opisan prema Palmeru (2002).

- **Razvijanje općeg modela**

U samom početku razvoja cjelokupnog modela, stručnjaci za domenu su već upućeni u opseg, kontekst i zahtjeve sistema za izgradnju (Abrahamsson *et al*., 2002). Dokumentirani zahtjevi često su dio ove faze. Međutim, FDD ne rješava eksplicitno pitanje prikupljanja i upravljanja zahtjevima.

Poslije svakog detaljnog pregleda, razvojni tim radi u malim grupama kako bi proizveo objektne modele za domen. Zatim, on diskutuje i odlučuje o odgovarajućim objektnim modelima za svako područje domene. Istovremeno, za sistem se sklapa ukupni oblik modela (Palmer i Felsing 2002).

- **Pravljenje liste karakteristika**

Dobar temelj za izgradnju sveobuhvatne liste karakteristika za sistem mogu omogućiti priručnici, objektni modeli i postojeća dokumentacija o potrebama. Razvojni tim na listi predstavlja svaku od vrijednosnih funkcionalnosti koje trebaju biti dio sistema. Korisnici i sponzori sistema provjeravaju listu funkcija zbog njihove validnosti i potpunosti.

- **Planiranje po funkcionalnostima**

Ovo podrazumijeva krojenje plana na visokom nivou, u kojem se skupovi funkcija nizaju prema prioritetu i ovisnostima, te se dodjeljuju glavnim programerima. Identificirane klase u procesu "razvoja cjelokupnog modela" dodjeljuju se određenim programerima tj. vlasnicima klasa.

- **Dizajn i izgradnja po funkcionalnosti**

Mala skupina funkcionalnosti odabrana je iz skupova karakteristika, a vlasnici klase kreiraju timove neophodne za razvoj odabranih funkcionalnosti. Ovaj proces je iterativna procedura kojom se isporučuju određene funkcionalnosti. Trajanje jedne iteracije je od nekoliko dana do najviše dvije sedmice. Ovaj proces uključuje aktivnosti kao što su: inspekcije koda i dizajna, jediničnog testiranja, integracija i kodiranja. Kada se iteracija uspješno završi, dovršene funkcionalnosti se promoviraju u glavnu građu, dok iteracija dizajniranja i izrade počinje novom grupom funkcionalnosti preuzetom iz skupa karakteristika (vidi Sliku 5).

3.2.2.2. Uloge i odgovornosti

Uloge su podijeljene u tri kategorije: ključne, sporedne i dodatne uloge (Palmer i Felsing 2002). Šest ključnih uloga u FDD metodi su: menadžer projekta, glavni arhitekta, menadžer razvoja, glavni programer, vlasnik klase i stručnjaci za domenu. Pet pomoćnih uloga sadrže menadžera izdanja (engl. release manager), jezičkog advokata, inženjera izgradnje, majstora alata i administratora sistema. Tri dodatne uloge koje su potrebne u bilo kojem projektu su ispitivači, implementatori i tehnički pisci. Jedan član tima može imati više uloga, a isto tako jednu ulogu može imati nekoliko osoba (Abrahamsson *et al*., 2002).

U daljnjem tekstu se nalazi obrazloženje uloga i odgovornosti, kako ih je predstavio Palmer (2002).

- **Menadžer projekta-** Osoba koja je administrativni i finansijski rukovodilac projekta. Njegova odgovornost je zaštititi projektni tim od vanjskih negativnih uticaja i omogućit mu rad uz odgovarajuće radne uvjete. U FDD metodologiji, menadžer projekta vodi glavnu riječ o opsegu, rasporedu i članovima projekta.
- **Glavni arhitekta-** Član tima koji je odgovoran za sveukupan dizajn sistema i vođenje radionica dizajna koje održava s timom (Palmer i Felsing 2002). Također, finalne odluke o svim pitanjima dizajna donosi isključivo on.
- **Menadžer razvoja-** On održava svakodnevne razvojne aktivnosti i rješava potencijalna nerazumijevanja unutar tima. Pored toga, menadžer rješava probleme sa resursima.
- **Glavni programer-** Osoba koja je iskusan programer, koja pri tom sudjeluje u analizi zahtjeva i dizajnu projekata. On preuzima odgovornost za vođenje malih timova u

razvoju novih funkcija analizi i dizajnu. Izbor funkcionalnosti koje će se isporučiti u sljedećoj iteraciji procesa "dizajn i izgradnja po funkcionalnosti" i identificiranje klasa i vlasnika klasa koji su neophodni timu, saradnja s drugim glavnim programerima u rješavanju tehničkih i resursnih pitanja, te sedmično izvješće o napretku tima.

- **Vlasnik klase-** Ova uloga je pod vodstvom glavnog programera u aktivnostima poput projekovanja, kodiranja, testiranja i dokumentiranja. On je dodijeljen klasi kao vlasnik iste, te je za nju zaslužan. Također, on formira timove po funkcionalnostima.
- **Stručnjak za domenu-** Stručnjak za domenu može biti korisnik, sponzor, klijent, poslovni analitičar ili neka kombinacija. On treba vladati znanjem o tome kako trebaju djelovati različiti zahtjevi za sistem tokom razvoja. Dužan je prenijeti svoje znanje programerima kako bi programeri bili u mogućnosti da isporuče kompetentan sistem.
- **Menadžer domene-** On je nadređen stručnjaku za domene i rješava njihove nejasnoće u pogledu zahtjeva za sistem.
- **Menadžer izdanja-** On je dužan da kontroliše napredak procesa analizirajući izvještaje o napretku glavnih programera i održava sa njima kratke sastanke o napretku. Također, dužan je da izvještava voditelja projekta o napretku.
- **Jezički pravnik-** Osoba koja treba posjedovati krucijalno znanje, kao što je poznavanje programskog jezika ili tehnologije. Posebna značajnost ove uloge jeste kada projekat uvodi neku novu tehnologiju.
- **Inženjer za izgradnju-** U njegovoj je dužnosti postavljanje, održavanje i pokretanje procesa izgradnje. Tu se podrazumijevaju i zadaci upravljanja sistemom kontrole verzija i objavljivanja dokumentacije.
- **Administrator sistema-** Njegov zadatak je konfiguriranje, upravljanje i rješavanje problema sa serverima, mrežom radnih stanica, testnim i razvojnim okruženjima. On može biti dio proizvodnje sistema koji se razvija.
- **Tester-** Ispitivači testiraju da li će isporučeni sistem zadovoljiti zahtjeve kupca.
- **Deplojer-** Njegov zadatak jeste konvertovanje postojećih podataka u format koji traži novi sistem i dio je implementacije novih izdanja.
- **Tehnički pisac-** Korisničku dokumentaciju spremaju tehnički pisci koji mogu formirati nezavisni tim ili biti dio projektnog tima.

3.2.2.3. Prakse

FDD fokusira se na skup "najbolje kombinovanih praksi". Iako one nisu nove, specifična kombinacija istih čine FDD proces jedinstvenim, gdje ni jedna praksa ne dominira cijelim procesom (Palmer i Felsing, 2002). Te prakse su sljedeće (Abrahamsson *et al*., 2002):

- **Objektno modeliranje domene**: Istraživanje i objašnjenje domene problema. Rezultati se dobijaju u okviru u koji se dodaju funkcionalnosti.
- **Razvoj po funkcionalnostima:** Razvoj i isporuka kroz male dijelove sadržane od malih funkcionalno razloženih funkcija.

- **Individualno vlasništvo nad klasom (kodom):** Svaka klasa sadrži po jednu osobu kojoj je dodijeljena odgovornost za njene performanse, dosljednost i konceptualni integritet klase.
- **Timovi vezani za funkcionalnosti:** Uz svaku funkcionalnost je jedan mali, dinamički formirani tim.
- **Inspekcija:** Tokom razvoja se upotrebljavaju najpoznatiji mehanizama za detekciju nedostataka.
- **Stalna izgradnja:** Odnosi se na to da se osigura konstantan razvoj sistema koji se uvijek može demonstrirati. Osnov čine redovne verzije na koje bi se mogle dodati nove funkcije.
- **Upravljanje konfiguracijom:** Pruža identifikaciju i historijsko praćenje posljednjih verzija svakog završenog fajla izvornog koda.
- **Izvještavanje o napretku:** Izvještaj o napretku vrši se na osnovu cjelokupnog rada na svim organizacionim nivoima.

Sve prethodno navedene prakse bi se trebale primijeniti od strane projektnog tima kako bi bio u skladu s razvojnim pravilima FDD-a. Međutim, daje se prostor projektnom timu da ih prilagodi prema svom iskustvu.

3.2.3. Kanban

Sve je počelo početkom 1940-ih. Prvi Kanban sistem razvio je Taiichi Ohno (industrijski inženjer i biznismen) za Toyotinu automobilsku industriju u Japanu. Stvoren je kao jednostavan sistem planiranja, čiji je cilj bio optimalno kontrolirati i upravljati radom i ograničiti količinu zaliha u svakoj fazi proizvodnje (Digite, 2013).

Dok je Taiban Ohno Kanban predstavio u proizvodnoj industriji, David J. Anderson je prvi koji je koncept primijenio na IT, razvoj softvera i općenito rad na znanju 2004. godine.

Kanban (u prijevodu „vizuelna karta“, „tabla“ ili „bilbord“) je neometani (engl. non-disruptive) evolucijski sistem upravljanja promjenama. To znači da se postojeći proces poboljšava u malim koracima. Primjenom mnogih manjih promjena (umjesto velikih) smanjuje se rizik za cjelokupni sistem. Evolucijski pristup Kanbana dovodi do slabog ili nikakvog otpora u timu i uključenim sudionicima. Svrha Kanbana je kontinuirano poboljšavati vlastiti proces rada (Digite, 2013).

Kanban je korišten u Microsoftovim operacijama za razvoj softvera 2004.godine. Od tada je Kanban s oduševljenjem usvojen u IT, Ops, DevOps i aplikacijskim / softverskim timovima.

Posebnost Kanbana je u tome što se može primijeniti na bilo koji postupak ili metodologiju. Bez obzira da li neko već koristi agilnu metodu kao što je Scrum, XP i druge, ili više tradicionalnih metoda - vodopad, iteraciju itd. - na to se može primijeniti

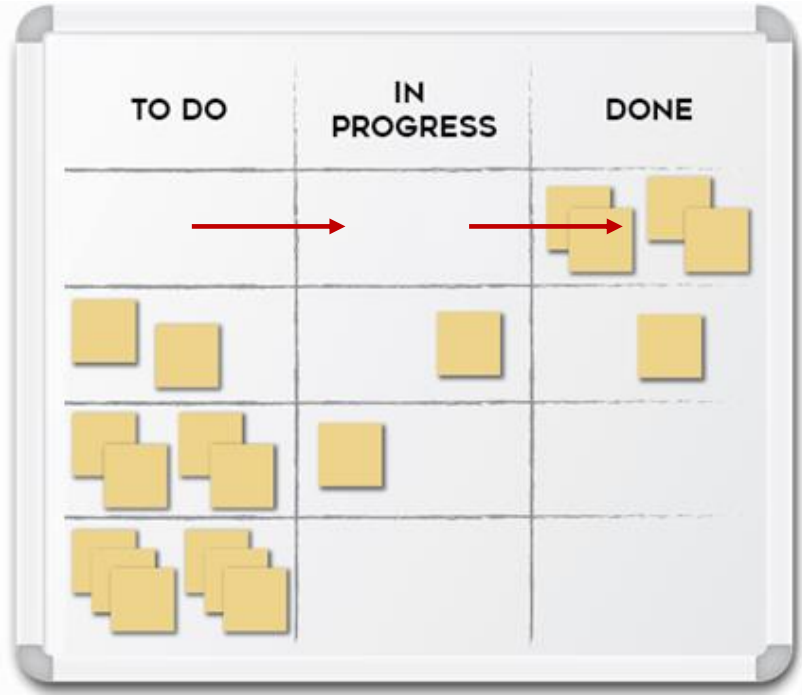
Kanban kako bi se postupno počeo poboljšavat proces, smanjivalo vrijeme ciklusa i poboljšao protok. U tom procesu naći će se na putu kontinuirane isporuke karakteristika, proizvoda ili usluga.

Vrijednosti koje zagovara kanban su: *transparentnost, fokus na korisnike, balans, rad kroz tok, međusobno razumijevanje, liderstvo, dogovaranje, poštovanje*.

3.2.3.1. Proces

Prvi korak u uvođenju Kanbana je vizualizacija procesa rada. To se radi u obliku Kanban ploče koja se sastoji od jednostavne bijele ploče i ljepljivih bilješki ili kartica. Svaka karta na ploči predstavlja zadatak (Digite, 2013). Transparentnost koju ovo stvara također doprinosi kulturnim promjenama (Kniberg *et al*., 2010).

Slika 6 Kanban tabla



Izvor: Doshi (2018)

U klasičnom modelu ploče Kanban postoje tri kolone, kao što su prikazane na slici:

(1) „To Do“: Ova kolona navodi zadatke koji još nisu započeti (zvani "backlog").
(2) „Doing“: Sastoji se od zadataka koji su u toku.
(3) „Done“: Sastoji se od zadataka koji su izvršeni.

U srži Kanbana je koncept „protoka“. To znači da bi kartice trebale prolaziti kroz sistem što ravnomjernije, bez dugog čekanja ili blokada. Budući da je WIP (radu u toku) ograničen u Kanbanovom sistemu, sve što postane blokirano iz bilo kojeg razloga teži da

začepi sistem. Ako se blokira dovoljno radnih predmeta, cijeli proces se zaustavlja. To ima za posljedicu fokusiranje cijelog tima i šire organizacije na rješavanje problema, deblokadu predmeta i obnavljanje protoka (Kniberg *et al*., 2010). Kanban ima različite tehnike, metrike i modele, a ako se oni dosljedno primjenjuju, to može dovesti do kulture kontinuiranog usavršavanja (kaizen) (Digite, 2013).

S tim da radne stavke teku kroz kanban sistem u jednom komadu, a svaki sistem je različit zbog faza u svom toku rada, najbolji način da se predstavi životni ciklus Kanbana jeste kroz uključene petlje povratnih informacija, a one su (Agile Alliance, 2021):

Pregled strategije (kvartalno)- Biraju se usluge koje će se pružati i kontekst u kojem su te usluge prikladne.

Pregled operacija (mjesečno)- Pravi se ravnoteža među uslugama, uključujući raspoređivanje ljudi i resursa kako bi se isporučila maksimalna vrijednost.

Pregled rizika (mjesečno)- Ispravno razumijevanje i adekvatno odgovoravanje na rizike isporuke usluge.

Pregled pružanja usluga (dvotjedno)- Ispitivanje i poboljšanj efikasnosti usluge. Ovo je poput retrospektive koja je bazirana na poboljšanje kanban sistema.

Sastanak za dopunu (sedmično)- Identifikacija stavki na kojima će tim raditi i određivanje koje se mogu odabrati sljedeće. Ovo je analogno sastanku za planiranje sprinta ili iteracije.

Kanbanski sastanak (svakodnevno)- Tim koji radi na usluzi koordinira njihove aktivnosti za taj dan. Ovo je analogno svakodnevnom zaustavljanju.

Sastanak o planiranju isporuke (po dolasku faze isporuke)- Praćenje i planiranje isporuke kupcima.

3.2.3.2. Uloge i odgovornosti

Uloge koje su se najčešće prepoznale u praksi i koje služe određenim svrhama su osobe koje već rade u nekim od postojećih funkcija u kompaniji (Agile Alliance, 2021).

- **Menadžer zahtjeva za uslugu**

Osoba koja treba da razumije potrebe i očekivanja kupaca i olakšava odabir i raspoređivanje funkcionalnosti na sastanku za dopunu. Ovu funkciju često obavlja menadžer proizvoda, vlasnik proizvoda ili menadžer usluga.

- **Menadžer pružanja usluga**

Osoba koja je nadležna za tok posla za isporuku odabranih predmeta klijentima. Također, on priprema sastanak i planiranje isporuke u Kanbanu. Drugi nazivi za ovu funkciju su voditelj protoka, voditelj isporuke ili master protoka.

3.2.3.3. Prakse

Kanban se zasniva na 6 osnovnih praksi koje su od suštinskog značaja (Agile Alliance, 2021).

- **Vizualizacija onoga što se radi danas (toka rada)**

Kanban sistemi se služi mehanizmom kanban ploče za vizualizaciju posla i procesa koji se radi. Da bi vizualizacija postigla veliku efikasnost, ona treba prikazati:

  - Tačku posvećenosti- gdje tokom procesa tim koji radi na proizvodu sklapa dogovor da radi na konkretnom predmetu
  - Tačku isporuke- gdje tim isporučuje proizvod kupcu
  - Pravila koja određuju koji bi zadatak trebao postojati u određenoj fazi
  - Količinu rada u toku (engl. Work in progress-WIP) ograničenja

- **Ograničenje WIP-a (količina rada u toku)**

Kada se uspostave ograničenja na količinu posla koji je u toku, ta nam ograničenja služe za usmjerenja pri dolasku novih stavki, te se time balansira tok posla da ne bi bilo previše rada istovremeno u jednoj fazi, smanjuje se vrijeme izvođenja, poboljšava kvalitet, te dolazi do češće isporuke.

- **Poboljšanje protokom**

Tok rada teži da maksimizira isporuku vrijednosti, smanji vrijeme izvođenja i da je predvidljiv. Timovi kroz transparentnost, inspekciju i adaptaciju uravnotežuju potencijalno sukobljene ciljeve. Krucijalni način upravljanja protokom je identifikovanje i rješavanje svih blokatora toka. Važno je naglasiti da kada se jedna stavka završi to otvara prostor u toku za novu stavku.

- **Stvoriti politike procesa eksplicitnim**

Eksplicitne politike pomažu u objašnjavanju procesa izvan samo popisa različitih faza u toku rada. Politike trebaju biti oskudne, jednostavne, dobro definirane, vidljive, uvijek primijenjene i lako promjenjive od strane ljudi koji rade na usluzi. Primjeri politika uključuju: WIP ograničenja, raspodjelu kapaciteta, definiciju gotovog i druga pravila za radne predmete koji postoje u različitim fazama procesa.

- **Koristiti petlje povratnih informacija**

Petlje povratnih informacija (engl. feedbackloops) su značajni dijelovi svakog sistema koji želi stvoriti evolucijske promjene.

- **Poboljšati suradnju, razvijati se eksperimentalno**

Kanban kreće sa postupkom kakav se trenutno nalazi, a potom aplicira kontinuirano i postupno poboljšanje umjesto pokušaja dostizanja unaprijed definiranog gotovog cilja.

3.2.4. „Crystal“ razvojni okvir

„Crystal“ razvojni okvir je metoda razvijena od strane američkog naučnika Alistair Cockburn-a, zaposlenika IBM-a. Njegov fokus nije bio na „korak-po-korak“ razvojnim strategijama, već na razvoju timske kooperacije i komunikacije (Kumar, 2020).

Ovaj agilni okvir naglašava pojedince i interakcije. On obuhvata niz različitih metodologija i bira se najprikladnija za svaki pojedinačni projekt, ovisno o veličini i kompleksnosti istog. Također uključuje i principe za prilagođavanje metodologija kako bi odgovarale različitim okolnostima različitih projekata (Abrahamsson *et al*., 2002).

Svaki član porodice Crystal označen je bojom koja predstavlja „težinu“ metodologije, gdje tamnija boja označava težu metodologiju. Na osnovu veličine i kritičnosti projekta se odabire odgovarajuća metodologija (Slika 7). U principu, veći projekti uglavnom zahtijevaju veću koordinaciju i potrebnija je teža metodologija. Također, što je kritičniji projekat, odabire se rigoroznija metodologija.

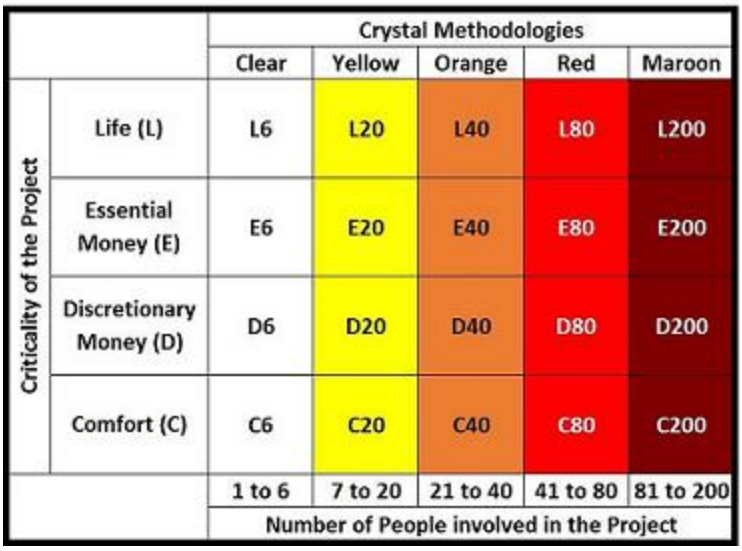
Na slici u nastavku nalaze se simboli znakova koji označavaju potencijalni gubitak izazvanim greškom sistema (tj. nivo kritičnosti): Udobnost/ **C**omfort (C), Diskrecioni novac/ **D**iscretionary money (D), Osnovni novac/ **E**ssential money (E) i Život/ **L**ife (L) (Abrahamsson *et al*., 2002).

Ukratko, nivo kritičnosti C pokazuje da pad sistema zbog grešaka uzrokuje gubitak udobnosti za korisnika, dok kvarovi u sistemu kritičnom za život mogu doslovno uzrokovati gubitak života.

Dimenzije kritičnosti i veličine prikazane su simbolom kategorije projekta opisanom na Slici 7; tako, na primjer, D6 predstavlja projekat sa najviše šest osoba koji isporučuje sistem maksimalne kritičnosti diskrecionog novca.

Određena pravila, karakteristike i vrijednosti su zajedničke Crystal metodologijama. Prvenstveno projekti primjenjuju inkrementalni razvoj sa maksimalnim trajanjem inkrementa do četiri mjeseca (Abrahamsson *et al*., 2002).

Slika 2 Dimenzije Crystal metodologije



Izvor: Arvin (2021)

Nazivi metoda su dobijeni po geološkim kristalima, a tri najpoznatije su crystal clear, crystal orange, crystal orange web. Prva dva su eksperimentisana u praksi i stoga su također predstavljena i razmatrana u ovom odjeljku (Abrahamsson *et al*., 2002).

3.2.4.1. Proces

Sve Crystal metodologije sadrže smjernice o standardima politike, proizvodima za rad, „lokalnim pitanjima“, alatima, standardima i ulogama koje treba primijeniti u razvojnom procesu. U nastavku su objašnjene Crystal Clear i Crystal Orange kroz njihov kontekst, sličnosti i razlike te ilustrirajući proces i aktivnosti Crystal Orange (Abrahamsson *et al*., 2002).

Crystal Clear je smišljena za vrlo male projekte (projekti kategorije D6), koji sadrže šest programera. Međutim, ako bi se proširila komunikacije i testiranja, primjenjiva je i na E8/D10 projekte. Tim ljudi bi trebao biti u zajedničkom poslovnom prostoru zbog limitacije u komunikacijskoj strukturi (Cockburn 2002a).

Crystal Orange je namijenjena za projekte srednje veličine, sa ukupno 10 do 40 članova projekta (kategorija D40), i sa vremenskim okvirom projekta od jedne do dvije godine. Također, projekti kategorije E50 mogu biti u okviru Crystal Orange s dodacima u njegove procese verifikacije-testiranja. U ovoj metodologiji, projekt je podijeljen na nekoliko timova s međufunkcionalnim grupama koristeći strategiju „Holistic Diversity“. Ipak, Crystal Orange ne podržava distribuirano razvojno okruženje, ali naglašava važnost vremena izlaska na tržište.

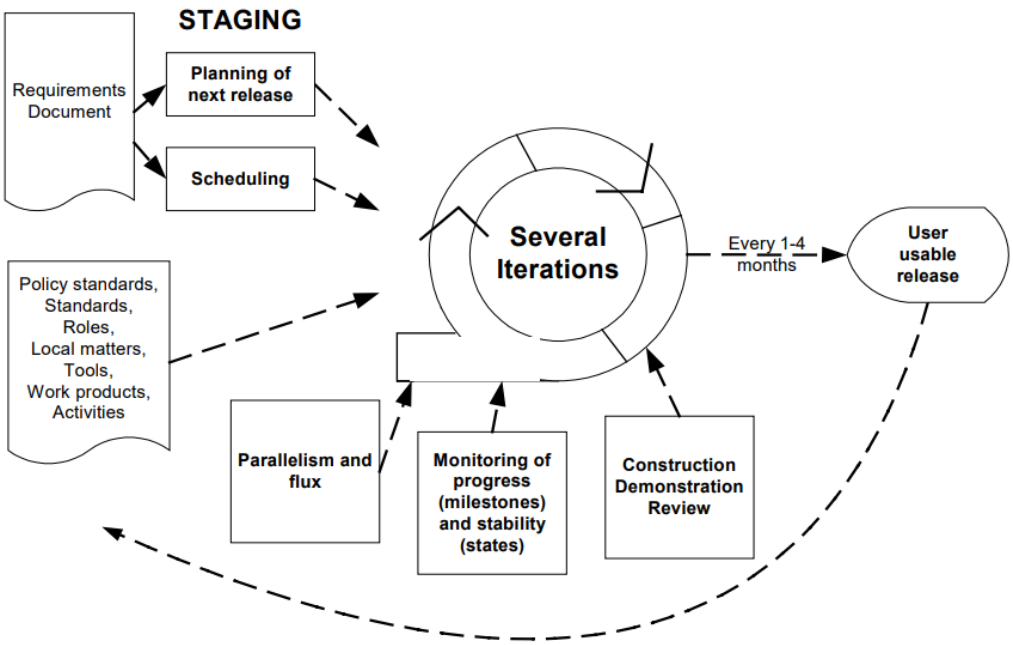
Standardi politike se primjenjuju tokom procesa. Zajednički standardi Crystal Clear i Crystal Orange su (Abrahamsson *et al*., 2002):

- Redovna postepena (engl. incremental) isporuka;
- Praćenje napretka prema prekretnicama kroz isporuke softvera i važnih odluka, a ne pisanih dokumenata;
- Direktno učešće korisnika;
- Automatsko testiranje funkcionalnosti;
- Dva korisnika pregledavaju po izdanju (engl. release);
- Radionice za podešavanje proizvoda i metodologije na početku i u sredini svakog koraka.

Jedina razlika između standarda politike ove dvije metodologije je ta što Crystal Clear predlaže postepenu isporuku u roku od dva do tri mjeseca, dok se u Crystal Orange inkrement može produžiti na najviše četiri mjeseca.

Standardi politike ovih Crystal metodologija su obavezni, ali se, međutim, mogu zamijeniti ekvivalentnim praksama drugih metodologija kao što su XP ili Scrum (Cockburn 2002a).

Slika 83 Jedan Crystal Orange Inkrement



Izvor: Abrahamsson et al. (2002)

3.2.4.2. Uloge i odgovornosti

Ovaj podnaslov govori nešto više o Crystal Clear i Crystal Orange ulogama prema Cockburnu (2002a). Krucijalna razlika između Crystal Clear i Orange jeste to što Crystal

Clear ima samo jedan tim u projektu. S druge strane, Crystal Orange sadrži više timova koje treba pratiti kroz projekt. Zajednička strana im je da jedan posao može raditi više uloga (Abrahamsson *et al*., 2002).

Glave uloge Crystal Clear metode su: sponzor, senior dizajner-programer, dizajner-programer i korisnik (Singh, 2021). Ove uloge sadrže više pod-uloga. Na primjer, dizajner-programer sastoji se od dizajnera poslovne klase, programera, softverskog dokumentariste i testera jedinica. Pod-uloge koje se mogu dodijeliti drugim ulogama su koordinator, poslovni stručnjak i sakupljač zahtjeva (Cockburn 2002a). Poslovni stručnjak je kao poslovni pogled u projektu, posjedujući znanje o specifičnom poslovnom kontekstu. Dužan je raditi na poslovnom planu, prateći stabilnost i moguće promjene istog (Cockburn 1998).

Pored Crystal Clear uloga, različite uloge su dio i Crystal Orange metode. One su podijeljene u nekoliko timova, kao npr. planiranja sistema, mentorstva projekta, arhitekture, tehnologije, funkcija, infrastrukture i eksternih timova za testiranje (Cockburn 2002a). Dalja podjela timovi jeste na međufunkcionalne grupe s različitim ulogama.

Crystal Orange prisvaja nekoliko novih uloga (Cockburn 1998; Cockburn 2002a), poput dizajnera korisničkog interfejsa, dizajnera baze podataka, stručnjaka za upotrebu, tehničkog facilitatora, arhitekte, poslovnog analitičara/dizajnera, mentora dizajna, tačke ponovne upotrebe (engl. rescue point), pisca i testera.

Crystal Orange (Cockburn 1998) sadrži vještine i tehnike potrebne za uspjeh u ovim ulogama. Jedna osoba u projektu može posjedovati više uloga. „Rescue point“ se odnosi na identifikaciju softverskih komponenti za višekratnu upotrebu i nabavku komercijalnih komponenti. Uloga pisca podrazumijeva izgradnju vanjske dokumentacije, kao što je specifikacija ekrana i korisničkih priručnika (Cockburn 1998). Dizajner poslovnih analitičara komunicira i pregovara s korisnicima kako bi specificirao zahtjeve i zaslon, te pregledao dizajn.

Ono što svaka grupa najmanje sadrži jeste dizajner poslovne analize, dva do tri dizajnera-programera, UI dizajner, dizajner baze podataka i uglavnom i tester. Dodatno, tehnički facilitatori mogu biti neophodani članovi grupe. Razlog za postojanje međufunkcionalnih grupa je smanjenje isporučivih proizvoda i poboljšanje lokalne komunikacije (Abrahamsson *et al*., 2002).

3.2.4.3. Prakse

Brojne prakse poput inkrementalnog razvoja su dio Crystal metodologije. U opisu Crystal Orange (Cockburn 1998), inkrement sadrži aktivnosti poput izvođenje (engl. staging), nadgledanja, pregledavanje. Ostale aktivnosti i prakse su navedene također u nastavku.

- **Staging**

Planiranje sljedećeg inkrementa događa se u ovoj fazi (Cockburn 1998). Pored toga, programeri planiraju i sljedeće izdanje. Urađen je raspored aktivnosti završenih do trećeg mjeseca (Cockburn 2002a). Osim toga, razvojni tim, koji bi na tome radio, odabrat će zahtjeve koje je potrebno implementirati u tom određenom inkrementu i koji su bili zakazani (Singh, 2021).

- **Revizija i pregled**

Svaki inkrement zahtijeva nekoliko iteracija, a svaka iteracija sadrži: izgradnju, demonstraciju i pregled ciljeva inkrementa (Abrahamsson *et al*., 2002).

- **Monitoring**

Napredak se prati u pogledu rezultata tima tokom razvojnog procesa s obzirom na njihov napredak i stabilnost (Cockburn 1998). Napredak se mjeri prekretnicama

Monitoring osigurava da napredak bude prema planu. Drugim riječima, to se odnosi na rezultate tima tokom kompletnog procesa razvoja softvera koji se odnosi na njihov rast i stabilnost (Singh, 2021). Kontinuirani napredak je mjerljiv kroz različite prekretnice kao što su: (početak, pregled 1, pregled 2, test, isporuka) i fazama stabilnosti (izrazito fluktuirajuća, fluktuirajuća i dovoljno stabilna za pregled). Nadgledanje je potrebno i u crystal clear i u kristalno orange (Abrahamsson *et al*., 2002).

- **Paralelnost i tok**

Paralelnost i protok znače istovremeno izvođenje dva djela. Drugim riječima, to znači da, kada nadzorni tim potvrdi da su svi rezultati dovoljno stabilni, započinje sljedeći zadatak. Sada, u ovoj fazi, većina timova može paralelno raditi na dodijeljenim im radovima (Singh, 2021).

- **Strategija holističke raznolikosti**

Budući da je Crystal Clear metoda koja je vrlo mala u veličini tima, nije joj potrebna strategija raznolikosti. Ostale crystal metodologije osim Crystal Clear slijede holističku strategiju raznolikosti (Singh, 2021).

To znači da se dijele veliki timovi u male grupe prema njihovim funkcionalnostima i stručnosti. Ideja iza ovoga je uvođenje više specijalnosti u jedan tim (Cockburn 1998).

- **Tehnika podešavanja metodologije**

Ovo je jedna od osnovnih tehnika pojednostavljivanja metoda za različite kristalne metode. Ova tehnika koristi podatke intervjua s projektima, radionica i povratnih informacija za otkrivanje ili definiranje nove tehnike ili za ispravno podešavanje bilo koje postojeće metodologije Crystal porodice. Cilj koji se želi postići jeste popraviti ili poboljšati trenutni razvojni proces (Singh, 2021).

Kristalna metoda teži da iz svakog inkrementa, projekt donese određeno znanje. Osim toga, također nam pomaže da saznamo koji je alat ili tehnika prikladniji za koju kristalnu metodu (Singh, 2021).

- **Pregledi korisnika**

Kristalne metode predlažu da za svako izdanje (release) krajnji proizvod korisnici pogledaju najmanje dva puta. Ovo smanjuje šanse greške provjere ili testiranja. Za velike projekte, ovaj broj korisničkog pregleda može se povećati na po tri inkrementa (Singh, 2021).

- **Radionice refleksije**

Većina Crystal metodologija za male projekte, poput Crystal Clears i Crystal Orange, ne definira nikakve posebne tehnike za korištenje u svojim projektima. Mogu usvojiti tehnike iz drugih popularnih metoda poput XP -a, Scrum -a i drugih. Osim toga, čak mogu i zamijeniti svoje tehnike (Singh, 2021).

Kristalne metode usvojile su tehniku poznatu kao refleksija. To podrazumijeva održavanje radionica utisaka prije i poslije svakog inkrementa. Štaviše, svrha održavanja ovih radionica je promicanje stalnog učenja iz rezultata testiranja, povratnih informacija i iskustava.

3.2.5. „Dynamic system“ razvojni okvir

Počeci razvojne metode dinamičkih sistema (skraćeno DSDM) doseže još do 1994. godine, nakon čega je postupno postala vodeći okvir za razvoj aplikacija (RAD) u UK (Stapleton, 1997). To je neprofitna i ne-vlasnička metoda za razvoj aplikacija, koji održava DSDM Konzorcij. Pored toga što je ona metoda u opće prihvaćenom smislu, DSDM pruža i okvir za kontrol razvoja aplikacija, upotpunjen smjernicama o efikasnom načinu korištenja te kontrole (Abrahamsson *et al*., 2002).

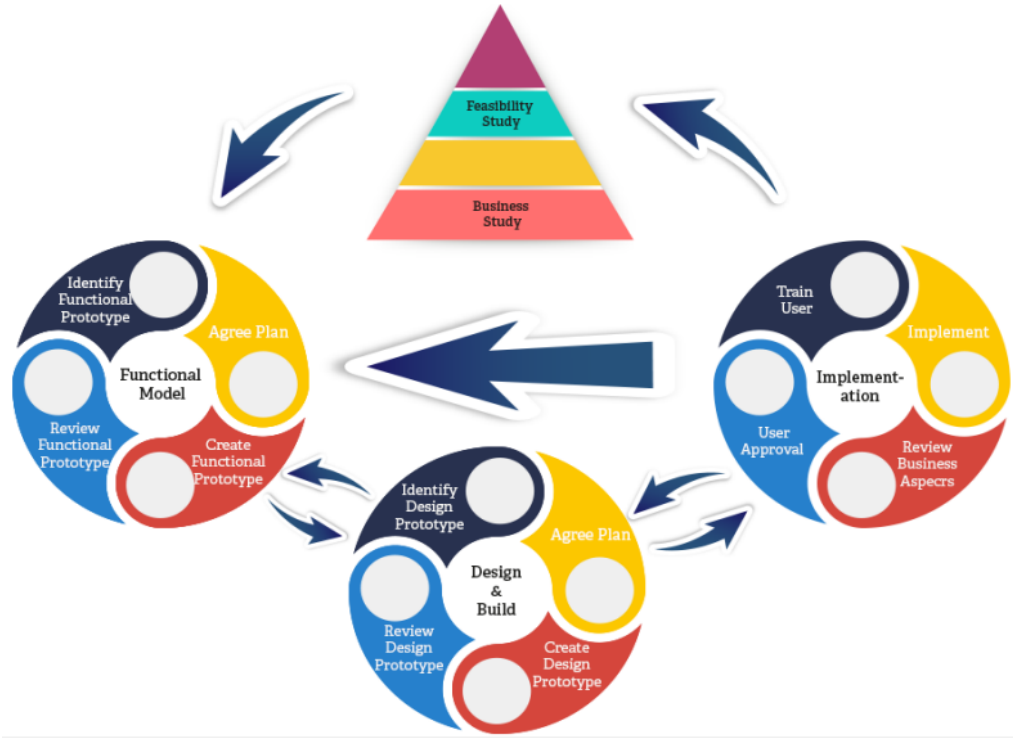
Osnovna ideja DSDM -a leži u popravljanju vremena i resursa, a potom da se prema tome prilagođava količina funkcionalnosti, prije nego li se radilo obratnom kao npr. prvo popravljalo količine funkcionalnosti u proizvodu, a potom prilagođavalo vrijeme i resurs u dostizanju te funkcionalnosti.

3.2.5.1. Proces

DSDM proces ima pet faza: studija izvodljivosti, studija poslovanja, iteracija funkcionalnog modela, projektovanje i iteracija izgradnje i implementacija (Slika 9). Prve dvije faze idu jedna za drugom i samo se jednom dešavaju. Međutim posljednje tri faze, koje su zadužene za konkretan razvoj, su iterativne i inkrementalne. Vremenski okvir traje onoliko koliko je unaprijed definiran vremenom, a iteracija mora završiti unutar

vremenskog okvira. Najčešće trajanje vremenskog okvira je od nekoliko dana do nekoliko sedmica (Abrahamsson *et al*., 2002).

Slika 9 Dijagram DSDM procesa



Izvor: Zafar et al. (2018)

Prva faza je **studij izvodljivosti** i u njoj se pravi procjena prikladnosti DSDM-a datom projektu, a to sve ovisi od vrste projekta, organizacijskih i kadrovskih pitanja. Pored toga, uzima se u obzir tehničke mogućnosti realizacije projekta, te njegovi rizici. Pripremaju se dva proizvoda rada - izvještaj o izvodljivosti i okvirni plan razvoja. Ne očekuje se da će studij izvodljivosti trajati više od nekoliko sedmica.

U sljedećoj fazi koja nosi naziv **poslovna studija** se analiziraju osnovne karakteristike poslovanja i tehnologija. Preporučuje se održavanje radionica na kojima se okupi dovoljan broj klijentovih stručnjaka koji razmatraju sve bitne aspekte sistema da bi se dogovorili o razvojnim prioritetima. Opisi procesa predstavljeni su u Definiciji poslovnog područja (engl. Business Area Definition) u odgovarajućem formatu (ER dijagrami, modeli poslovnih objekata itd.). Dva rezultata su dobijaju u ovoj fazi: definicija sistemske arhitekture i okvirni plan prototajpinga.

Prva iterativna i inkrementalna faza jeste faza **funkcionalnog modela**. U svakoj iteraciji planiraju se sadržaj i pristup za iteraciju, a rezultati te iteracije se analiziraju za daljnje iteracije. Vrši se i analiza i kodiranje; izrađuju se prototipovi, a iskustva dobijena na njima služe za poboljšanje modela analize. Funkcionalni model proizvodi se kao proizvod koji sadrži kod prototipa i modele analize. Testiranje je sastavnio dio ove faze.

Postoje još četiri etape ove faze. **Prioritizirane funkcije** su prioritetne liste funkcionalnosti koje se isporučuju na kraju iteracije. Dokumenti o pregledu **funkcionalnih prototipova prikupljaju utiske** korisnika o trenutnom inkrementu, koji služe kao dopuna za kasnije iteracije. Tu su također i **nefunkcionalni zahtjevi**, koji se uglavnom rješavaju u narednoj fazi. Jedan od važnih dokumenata u ovoj fazi je i a**naliza rizika daljnjeg razvoja**, jer će već od sljedeće faze (projektiranje i izrada iteracije) pa nadalje susretat se sa nekim od rizika koje će trebati na vrijeme riješiti.

Iteracija projektiranja i izgradnje je faza gdje je veći dio sistema uglavnom razvijen. Testirani sistem treba da rezultira barem minimalno dogovorenim skupom zahtjeva. Korisnici su dužni da pregledaju dizajn i funkcionalne prototipe, te se na osnovu njihovih komentara nastavlja daljni razvoj.

Proces se završava sa fazom **implementacije** u kojoj se sistem prebacuje iz razvojnog okruženja u stvarno proizvodno okruženje. Korisnici se obučavaju i sistem im se isporučuje. Pored toga, dostavlja se i korisnički priručnik i izvještaj o pregledu projekta. Izvještaj je sadržan od ishoda projekta, a na osnovu rezultata određuje se smjer daljnjeg razvoja. DSDM ističe četiri moguća pravca razvoja. Ako sistem ispunjava sve zahtjeve, dorada nije potrebna. S druge strane, ako se mnogo zahtjeva mora ostaviti po strani (na primjer, ako nisu uočeni dok se sistem nije razvijao), proces se cijeli može ponoviti. Ako se neka manje kritična funkcionalnost mora izostaviti, proces se može ponovo pokrenuti od faze ponavljanja funkcionalnog modela pa nadalje. Na kraju, ako se neki tehnički nedostaci ne mogu riješiti zbog vremenskih ograničenja, oni će se riješiti ponovnim ponavljanjem od faze projektiranja i izgradnje pa nadalje.

3.2.5.2. Uloge i odgovornosti

Ova metodologija obuhvata čak 15 uloga za korisnike i programere. Prema Stapleton-u (1997), najdominantnije od njih su navedene daljnjem tekstu.

Developeri i **senior developeri** su jedine uloge odogovorne za razvoj. O tome ko će biti senior developer ovisi od iskustva u zadacima koje programer obavlja. Ova titula također upućuje na nivo liderstva u timu. Uloge programera i senior programera uključuju svo razvojno osoblje, bilo analitičare, testere, programere ili dizajnere.

Tehnički koordinator preuzima odgovornost za tehničku kvalitetu i kontrolu projekta, te definira arhitekturu sistema.

Od korisničkih uloga najbitnija je **ambasador korisnik**. Dužnost mu je unošenje znanja korisničke zajednice u projekat i prenošenje informacija o napretku projekta drugim korisnicima. To će pružiti priliku da se primi odgovarajuća količina povratnih informacija korisnika. Zbog stepena težine da ambasador korisnik predstavi sva potrebna gledišta korisnika, dodana je dodatna uloga **korisnika savjetnika**. To može biti bilo koji korisnik

ili npr. IT osoblje ili finansijski revizori, koji predstavljaju važno gledište sa stanovišta projekta (Abrahamsson *et al*., 2002).

Vizionar je korisnik koji ima najprecizniju percepciju poslovnih ciljeva sistema i projekta. On je uglavnom osoba koja pokreće početničku ideju izgradnje sistema. U dužnosti mu je da osigura isticanje bitnih zahtjevi još na početku i da projekt teži odvijanju u pravom smjeru sa stanovišta tih zahtjeva.

Izvršni sponzor je osoba iz korisničke organizacije koja ima financijska ovlaštenja i odgovornosti. Zbog toga, on ima krajnju moć u donošenju odluka.

3.2.5.3. Prakse

Ova ideologija definisana je sa devet praksi koje su detaljnije obrazložene u tabeli ispod (Stapleton, 1997).

Tabela 2 Prakse DSDM-a

DSDM prakse	Opis
Aktivno uključivanje korisnika je imperativ.	Nekoliko obrazovanih korisnika mora biti prisutno tokom razvoja sistema kako bi se osigurala pravovremena i tačna povratna informacija.
DSDM timovi moraju biti ovlašteni za donošenje odluka.	Dugi procesi donošenja odluka ne mogu se tolerirati u brzim razvojnim ciklusima.
Fokus je na čestoj isporuci proizvoda.	Pogrešne odluke se mogu ispraviti ako je ciklus isporuke kratak i korisnici mogu dati tačne povratne informacije.
Prikladnost za poslovnu svrhu osnovni je kriterij za prihvaćanje rezultata.	Prije nego što se zadovolje osnovne poslovne potrebe sistema, ne treba težiti tehničkoj izvrsnosti u manje važnim područjima.
Iterativni i inkrementalni razvoj neophodan je za približavanje tačnom	Puštanjem sistema da se razvijaju kroz iterativni razvoj, greške se mogu rano otkriti

poslovnom rješenju.	i ispraviti.
Sve promjene tokom razvoja su reverzibilne.	Koristeći kratke iteracije i osiguravajući da se prethodna stanja u razvoju mogu vratiti, pogrešan put se može sigurno ispraviti.
Zahtjevi su postavljeni na visokom nivou.	Zamrzavanje osnovnih zahtjeva treba izvršiti samo na visokom nivou, kako bi se omogućilo da se detaljni zahtjevi promijene po potrebi.
Testiranje je integrirano tokom čitavog životnog ciklusa.	Programeri i korisnici bi trebali testirati svaku komponentu sistema potpuno tokom razvoja.
Suradnički i kooperativni pristup koji dijele svi akteri je od suštinskog značaja.	Izbor onoga što se isporučuje u sistemu i onoga što je izostavljeno uvijek je kompromis i zahtijeva zajednički dogovor.

Izvor: Stapleton (1997)

4. SCRUM METODOLOGIJA

Bogata historija i prva pojava termina 'Scrum' počinje od Takeuchijeovog i Nonakovog članka iz davne 1986. godine pod nazivom "The New New Product Development Game", (Hidalgo, 2019) u kojem je predstavljen prilagodljiv, brz i samoorganizirajući proces razvoja proizvoda koji potiče iz Japana (Schwaber i Beedle 2002). Scrum nije skraćenica, već izraz koji izvorno potiče od strategije u igri ragbija, gdje se odnosi na način ponovnog pokretanja igre uz timski rad, nakon slučajnog prekršaja ili kada je lopta izašla iz igre. (Schwaber i Beedle 2002).

Jeff Sutherland i njegov tim su 1993.godine u sklopu kompanije „Easel“ stvorili Scrum proces za razvoj softvera kombinirajući koncepte iz članka iz 1986. godine sa konceptima objektno orijentiranog razvoja, empirijske kontrole procesa, iterativnog i inkrementalnog razvoja, softverskih procesa i istraživanja produktivnosti i složenih adaptivnih sistema. Potom, 1995. godine, Ken Schwaber objavio je na „OOPSLA“ prvi rad o Scrumu (Schwaber 1995), te je od tada sa Sutherland-om, zajedno i odvojeno, izdao nekoliko publikacija specifičnih o Scrumu (Rubin, 2013).

Scrum je procesni okvir, razvijen za upravljanje procesom razvoja proizvoda. Prema definiciji Schwaber i Sutherland (2017) Scrum je okvir unutar kojeg ljudi mogu da adresiraju složene probleme, dok produktivnošću i kreativnošću isporučuju proizvode najviše moguće vrijednosti (Agile Alliance, 2021).

On je izvorno dizajniran za razvoj softvera, ali se također može koristiti u kompanijama koje trebaju implementirati procese upravljanja projektima poput reklamnih agencija, arhitektonskih projekata i banaka (Azanha *et al*., 2017). Schwaber (2004) objašnjava da Scrum nije predvidljiv proces, te se koristi u složenim poslovima u kojima se događaji ne mogu predvidjeti, a pruža okvir i skup **principa** i **praksi** koji čine sve transparentnim. To omogućava timu da ima tačan uvid u situaciju tokom cijelog projekta i, ako je potrebno, izvrši odgovarajuće promjene kako bi postigao svoje ciljeve.

Scrum je zasnovan na **vrijednostima** poštenja, otvorenosti, hrabrosti, poštovanja, usredotočenosti, povjerenja, osnaživanja i saradnje (Rubin, 2013).Scrum počiva na ideji empirizma koja tvrdi da znanje dolazi iz iskustva i iz donošenja odluka na osnovu poznatoga. Prema Schwaberu (2007), Scrum je podržan od tri stuba principa: **transparentnost, inspekcija,** te **prilagođavanje**.

Prvi stub, transparentnost, osigurava da svi aspekti relevantni za uspjeh procesa ostanu vidljivi i poznati, kako bi se osiguralo da rezultat bude u skladu s prethodno definiranim. Inspekcija, drugi stup, napravljena je s ciljem da se otkrije svaka neusklađenost koja može štetiti rezultatima tima. Prilagođavanje kao treći stup, u kojem se iz identifikacije kvara/greške izvršavaju prilagodbe u procesu, ili u procesnim materijalima, smanjujući vjerovatnoću lošeg ishoda. (Azanha *et al*., 2017)

4.1. Scrum proces

Proces kreće od **vlasnik proizvoda** koji ima viziju onoga što se treba napraviti. Sve karakteristike osmišljenog proizvoda se unose u **Product Backlog** listu. Budući da proizvod sam po sebi može biti kompleksan, kroz aktivnost koja se naziva „sređivanje“ (engl. grooming) stavke iz navedene liste će se poredati po prioritetima. Sprint (vremenski okvir za razvoj inkrementa) predstavljen je u ilustraciji velikom sivom strelicom na kojoj se nalaze Scum aktivnosti. Sprint kreće sa **planiranjem sprinta** gdje razvojni tim mora odrediti podskup stavki Product Backloga za koje vjeruje da ih može izgraditi.

Potom dolazi **sprint backlog** kojeg stvaraju članovi tima kako bi stekli povjerenje da je razvojni tim donio razumne odluke, te se ovdje stavke redaju po prioritetima. Između ostalog, ovdje je obrazloženo kako tim planira dizajnirati, izgraditi, integrirati i testirati odabrane karakteristike tokom tog sprinta. Nakon toga slijedi **sprint izvršenja**, gdje razvojni tim obavlja posao potreban za realizaciju željenog proizvoda. Uporedo sa tim, Scrum master pomaže timu da poveća produktivnosti, transparentnost i da lakše upravlja poslom kroz dnevni Scrum koji se obavlja svaki dan. Na kraju ovog događaja,

potencijalno isporučivi inkrement je proizveden od strane razvojnog tima, a on može da bude dio, ali i cijela vizija vlasnika proizvoda.

Scrum tim završava sprint izvodeći dvije aktivnosti. Prva jeste **pregled sprinta** gdje svi sudionici i Scrum tim pregledavaju proizvod koji je proizveden i gdje se skupljaju povratne informacije. Druga aktivnost je **retrospektiva sprinta** gdje Scrum tim analizira Scrum proces koji je sproveden za kreiranje proizvoda, te njihove lične doživljaje i mišljenja kako bi doprinijeli većoj efikasnosti u narednom sprintu. Ishod ovih aktivnosti su uglavnom prilagodbe koje će biti unesene u Product Backlog i nakon koji će krenuti novi sprint ciklus. U slučaju da je vizija Vlasnika proizvoda dostignuta još sa prvim sprintom, proizvedeni inkrement će se isporučiti kupcu (Rubin, 2013).

Slika u nastavku predstavlja Scrum proces kojeg sačinjavaju Scrum tim, događaji i artefakti.

Slika 10 Scrum Proces

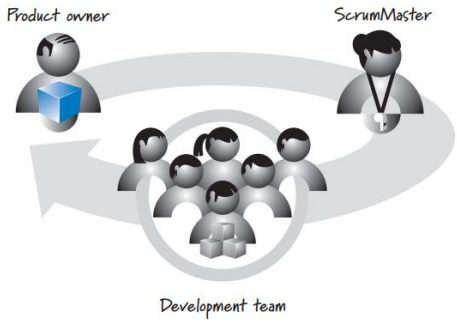
Izvor: Rubin (2013)

4.2. Uloge i odgovornosti

Ono što čini Scrum drugačijim od klasičnih metoda upravljanja projekta jeste to što, Scrum nema potrebu za vođom projekta, projektnim menadžerom ili vođom tima. Prema Schwaberu i Sutherlandu (2017), Scrum tim sastoji se od tri glavne uloge, koje imaju različite zadatke i ciljeve u procesu i praksi, a one su sljedeće: Scrum Master, Vlasnik proizvoda i Scrum razvojni tim (Slika 11). Svaka uloga u Scrum timu je jasno definisana i

svaka od njih posjeduje svoje odgovornosti. U nastavku rada su detaljnije objašnjene svaka od njih pojedinačno.

Slika 41 Scrum tim



Izvor: Rubin (2013)

4.2.1. Vlasnik proizvoda

Vlasnik proizvoda predstavlja zainteresiranu stranu odnosno *Stakeholder*-a ili glas korisnika i on je isključivo jedna osobu, a ne odbor (Schwaber i Sutherland, 2017). Odgovoran za upravljanje Backlogom proizvoda kako bi se postigao željeni ishod koji tim nastoji postići, te je odgovoran za maksimiziranje vrijednosti koju tim isporučuje (Agile Alliance, 2021).

On je jedini autoritet odgovoran za odlučivanje koje će karakteristike i funkcionalnosti graditi i kojim redoslijedom po prioritetima, na temelju važnosti i ovisnosti (Rubin, 2013). On je dužan da učini listu stavki vidljivom, transparentnom i jasnom svima kako bi se imao uvid u zadatke Scrum tima (Schwaber i Sutherland, 2017).

Vlasnik proizvoda održava i saopćava svim ostalim sudionicima jasnu viziju onoga što Scrum tim pokušava postići. Kao takav, vlasnik proizvoda odgovoran je za ukupni uspjeh rješenja koje se razvija ili održava. Kako bi osigurao da tim brzo izgradi ono što vlasnik proizvoda želi, vlasnik proizvoda aktivno surađuje sa Scrum Masterom i razvojnim timom, te mora biti dostupan na brzi odgovor za potencijalne nejasnoće (Rubin, 2013).

Komunikacija je ključna kompetencija vlasnika proizvoda, te je on kao takav dužan da premošćuje jaz u komunikaciji između tima i njegovih stakeholdera. U sklopu komunikacijskih zadataka on je dužan da predstavi rješenje ključnim stakeholderima i da ih informiše o razvojnom procesu, da organizuje preglede prekretnica i da pregovara o prioritetima, opsegu, rasporedu i finansiranju, te da najavi nove objave (Schwaber i Sutherland, 2017).

4.2.2. Scrum master

Za ispravno razumijevanje i primjenu Scrum metodologije odgovoran je Scrum Master. Scrum Master pomaže svima uključenima da razumiju i primijene Scrum vrijednosti, principe i prakse. On je poput trenera koji pomaže Scrum timu i ostatku organizacije da razviju vlastiti Scrum pristup visokih performansi, specifičan za organizaciju. U isto vrijeme, Scrum Master pomaže organizaciji kroz izazovan proces upravljanja, te je dužan da osigura da se uklone i premoste sve prepreke u procesu kako bi tim radio što produktivnije (Abrahamsson *et al*., 2002).

Scrum Master nema ovlasti da vrši kontrolu nad timom koji je samoorganizirajući, tako da ova uloga nije ista kao tradicionalna uloga projekt menadžera ili menadžera razvoja. Scrum Master funkcionira kao vođa, a ne kao menadžer (Rubin, 2013).

Kroz navedene aktivnosti on pomaže vlasnika proizvoda, razvojni tim i organizaciju (Schwaber i Sutherland, 2017).

- Pronalazi adekvatne tehnike za upravljanje Product Backlog stavkama;
- Pomaže Razvojnom timu da postane samoorganizirajući i višefunkcionalni, te da kreira proizvode visoke vrijednosti;
- Omogućuje razumijevanje i praktikovanje agilnosti;
- Omogućuje moderaciju Scrum događaja po potrebi ili po zahtjevu;
- Planira i implementira Scrum u organizaciji;
- Pomaže zaposlenima i zainteresovanim stranama da rade u skladu sa Scrum okvirom i empirijskim razvojem proizvoda;
- Upravlja promjenama koje povećavaju produktivnost Scrum tima;
- Radi sa drugim Scrum Master-ima na maksimiziranju učinkovitosti primjene Scrum-a u organizaciji.

4.2.3. Razvojni tim

Razvojni tim odgovoran je za isporuku gotovog proizvoda kojeg je vlasnik proizvoda tražio. Scrum-ove uloge razvojnog tima su raznolika, višefunkcionalna kolekcija disciplina (programer, UX/UI dizajner, tester, administrator baze podataka, arhitekta itd.) koje se sažimaju u jednu titulu koju obično nazivaju programerom odgovornim za dizajniranje, izgradnju i testiranju željenog proizvoda (Rubin, 2013).

Iako pojedini članovi razvojnog tima mogu imati specifične kompetencije i područja na koja se fokusiraju, odgovornost ostaje na razvojnom timu kao cjelini. Razvojni tim se samoorganizira kako bi odredio najbolji način za postizanje cilja Sprinta koji je postavio

vlasnik proizvoda, te niko (čak ni Scrum Master) ne može određivati kako će se iz Product Backlog stavke kreirati u potencijalno isporučivi inkrement (Schwaber i Sutherland, 2017). Optimalan broj članova razvojnog tima jeste od 3 do 9 osoba koje moraju kolektivno imati sve vještine potrebne za stvaranje kvalitetnog softvera (Rubin, 2013).

Scrum ne priznaje pod-timove Razvojnog tima, bez obzira na specifičnosti zadataka koji trebaju biti urađeni poput testiranja ili poslovne analize, te nema iznimki za ovo pravilo (Schwaber i Sutherland, 2017).

4.3. Prakse

Scrum prakse predstavljaju alate i radnje koji se odvijaju tokom Scrum procesa kako bi se proizveo jedan inkrement proizvoda. Te prakse sačinjene su od *Scrum artefekata* i *Scrum događaja* (Schwaber i Sutherland, 2017).

4.3.1. Scrum artefakti

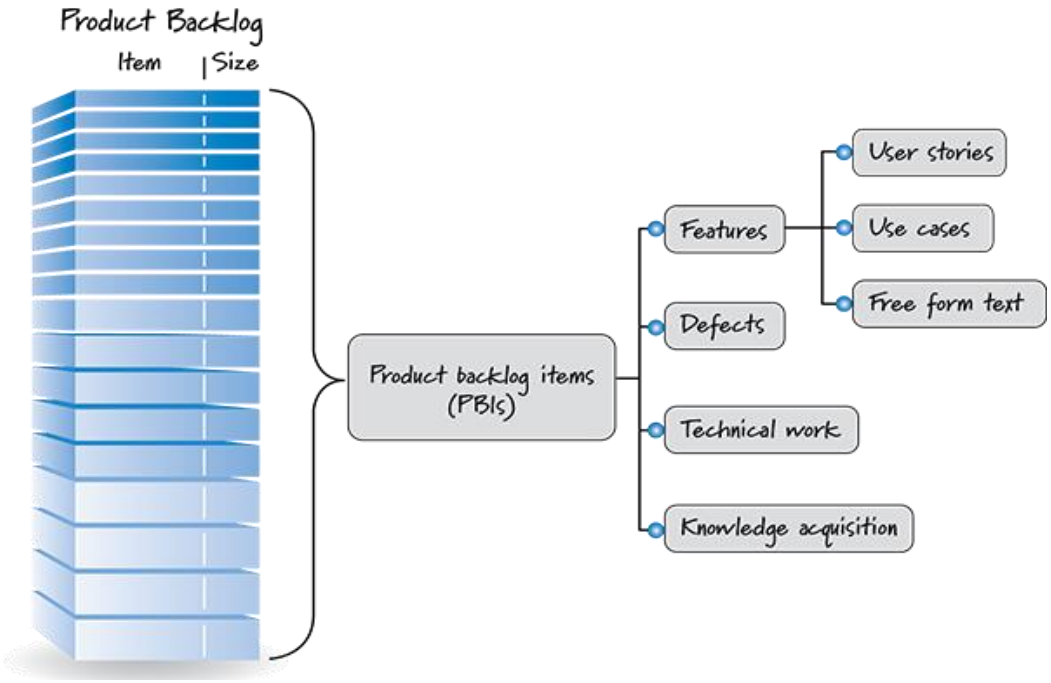
Scruma artefakti predstavljaju rad ili vrijednost na različite načine koji su korisni da osiguraju transparentnost i mogućnost za kontrolu i adaptaciju (Cervone, 2011; Schwaber and Sutherland, 2013). Artefakti definisani Scrum-om su posebno dizajnirani da maksimiziraju transparentnost ključnih informacija potrebnih da bi Scrum timovi bili uspješniji u isporuci "Done" inkrementa. U daljnjem tekstu će biti predstavljena tri Scrum artefakta, a to su: Backlog proizvoda, Backlog sprinta i inkrement.

4.3.1.1. Backlog proizvoda

Product Backlog definira sve što je potrebno u konačnom proizvodu na osnovu trenutnog znanja. To je prioritizirana i stalno ažurirana lista željenih funkcionalnosti, poslovnih i tehničkih zahtjeva za sistem koji se gradi ili poboljšava. Sačinjena je od Product Backlog stavki (skraćeno PBI), te je za nju odgovoran vlasnik proizvoda. Ona pruža zajedničko razumijevanje o tome šta će se i kojim redoslijedom praviti.

Product Backlog stavke mogu uključivati, na primjer svojstva, funkcije, ispravke grešaka, nedostatke, tražena poboljšanja i tehnološke nadogradnje. One se često pišu kao korisničke priče (ili se ponekad koriste slučajevi ili testovi slobodne forme). Slika ispod ilustruje Backlog Proizvoda (Rubin, 2013).

Slika 12 Backlog Proizvoda



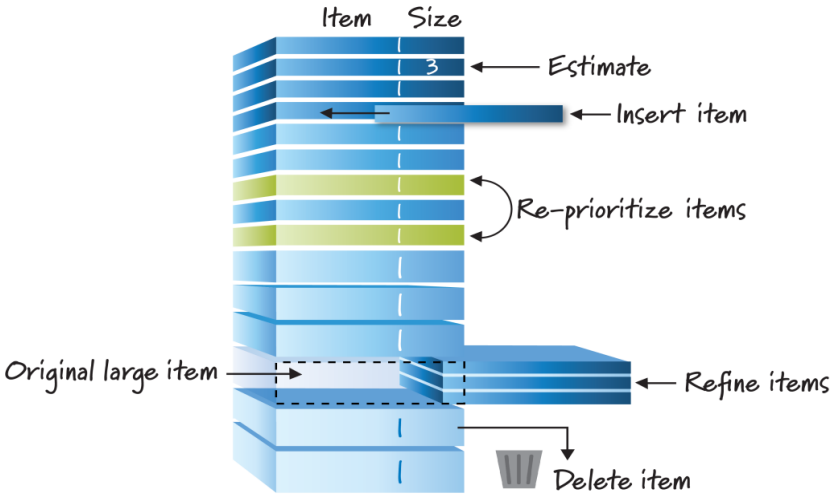
Izvor: Rubin (2013)

Product Backlog stavke se razlikuju po nivou detaljnosti. One sa kojima će se prvo raditi su na vrhu Backlog liste, te su visoko prioritetne i detaljne. Sa druge strane, one sa kojima se ne radi u skorije vrijeme nalaze se niže, manje su prioritetne i detaljne. U slučaju da želimo da poboljšamo neku stavku, one će rasti po prioritetu, dodajući joj detalje i vremenski okvir. Svakoj stavci se procjenjuje veličina radi procjene potrebnog napora za razvoj te stavke i prema tome se određuje i njen prioritet (Rubin, 2013).

Kako se proizvod razvija ili održava, tako se i ova lista ažurira. Što se više uči o proizvodu i tržištu, mogu se dodati nove stavke, odbaciti neke od postojećih ili ih promijeniti. Nagomilavanjem stavki je znak zdrave i funkcionalne liste, te znači da proizvod dobija na vrijednosti i da tržište pruža neke povratne informacije (Rubin, 2013).

„Grooming“ (u prijevodu „dotjerivanje“) je termin koji predstavlja sređivanje stavki u Product Backlogu. Sadržan je od tri glavne aktivnosti: kreiranje i prerada, procjena i određivanje prioriteta PBI-a (Slika 13). Navedene aktivnosti se odvijaju tokom cijelog razvoja proizvoda. Vlasnik proizvoda (donosilac odluka) je ovome najviše posvećen, ali značajno učešće je i od strane Scrum master-a, razvojnog tima i ključnih unutarnjih i vanjskih sudionika. U Scrum-u, „dotjerivanje“ stavki se može raditi više puta, ali je najčešće to tokom plana izdavanja, nakon pregleda sprinta, i kao regularni dio svakog sprinta (Rubin, 2013).

Slika 5 Uređivanje Backlog Proizvoda

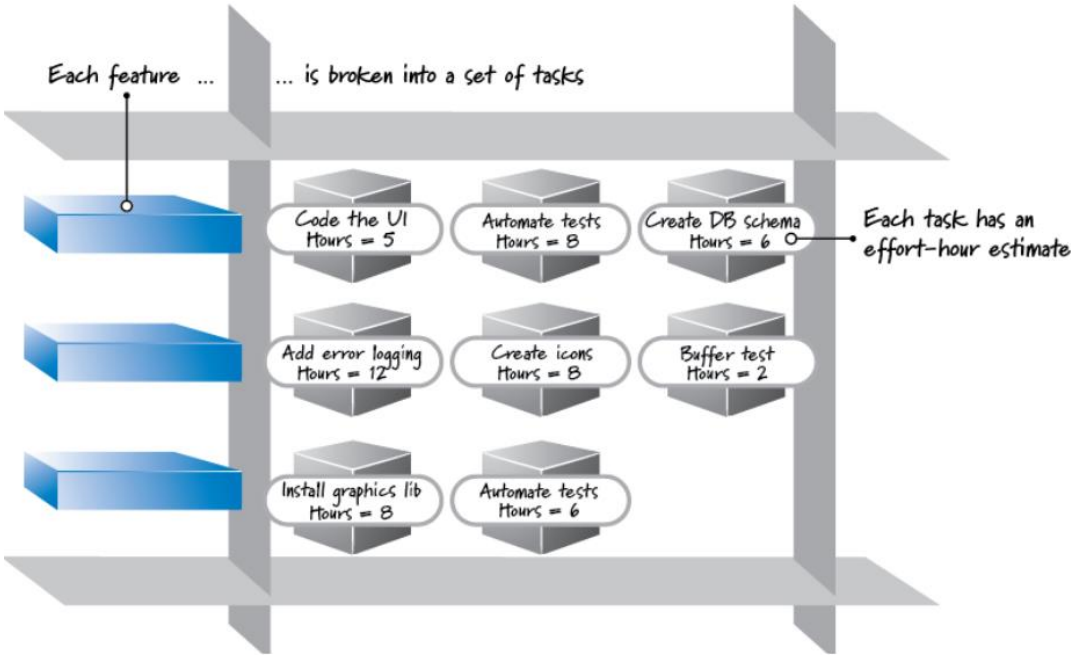


Izvor: Rubin (2013)

4.3.1.2. Backlog sprinta

Sprint Backlog je početna tačka za svaki Sprint (Abrahamsson *et al.*, 2002). To je lista stavki iz Product Backloga izabranih za implementaciju u sljedećem Sprintu kao dio plana za isporuku inkrementa i realizaciju cilja.

Slika 64 Sprint Backlog



Izvor: Rubin (2013)

Sprint Backlog je prognoza razvojnog tima o tome koliki rad je potreban da se funkcionalnosti implementiraju i dostave u "Done" inkrement. To je plan sa dovoljno detalja da bi promjene u procesu biti razumljive u dnevnom Scrum-u (Pogledaj 4.3.2.4.).

Razvojni tim modifikuje Sprint Backlog kroz Sprint i jedini je on ovlašten da ga mijenja.

Sprint Backlog stvara vidljivim sav posao koji razvojni tim identificira kao potreba da bi zadovoljio Sprintov cilj. Kako bi se osiguralo kontinuirano poboljšanje, ono podrazumijeva barem jedno poboljšanje procesa visokog prioriteta identificirano na prethodnom retrospektivnom sastanku (pogledaj 4.3.2.5.).

4.3.1.3. Inkrement

Inkrement predstavlja skup svih stavki iz Product Backlog-a koje su završene tokom jednog Sprinta, te vrijednost unaprijeđenja prethodnih Sprintova. Na kraju jednog Sprinta potencijalno isporučiv proizvod (inkrement) treba biti upotrebljiv i naznačen sa „Done". To je transparentan primjer onoga što je napravljeno vlastitim iskustvom učesnika, te je on jedan iskorak više ka viziji (Rubin, 2013; Schwaber i Sutherland, 2017).

Snaga ovog pristupa u kojem se izrađuju potencijalno isporučive verzije leži u povećanoj kontroli u strateškom planiranju daljnjeg razvoja u slučaju da kupac zahtijeva određene izmjene. Stoga, inkrement je poput prilike za kupaca da po potrebi unaprijedi proizvod dopunimskim zahtjevima (Rubin, 2013; Schwaber i Sutherland, 2017).

Slika 15 Potencijalno isporučiv proizvodni inkrement

Izvor: Rubin (2013)

Važno je naglasiti da termin „završenog" ili „Done" ne označava da se proizvod mora stvarno isporučiti, već da je on dovršen do tog nivoa da se može staviti u pogon. Uglavnom, najosnovnija definicija završenog predstavlja potpuni rad proizvoda (izdizajniran, iskodiran, integrisan, testiran i dokumentiran) - što bi pružilo sigurnu vrijednost kupcu (Rubin, 2013; Schwaber i Sutherland, 2017).

4.3.2. Scrum događaji

Azanha *et al*., (2017) tvrde da Scrum „zapošljava“ pet vrste događaja s fiksnom dužinom trajanja, koji se nazivaju i vremenski okviri, u cilju stvaranja regularnosti i manjoj učestalosti organiziranja sastanaka koji nisu unaprijed planirani. Pomenutih 5 događaja su:

(1) Sprint,
(2) Dnevni sastanak,
(3) Sastanak za planiranje Sprinta,
(4) Pregled Sprinta,
(5) Retrospektiva Sprinta.

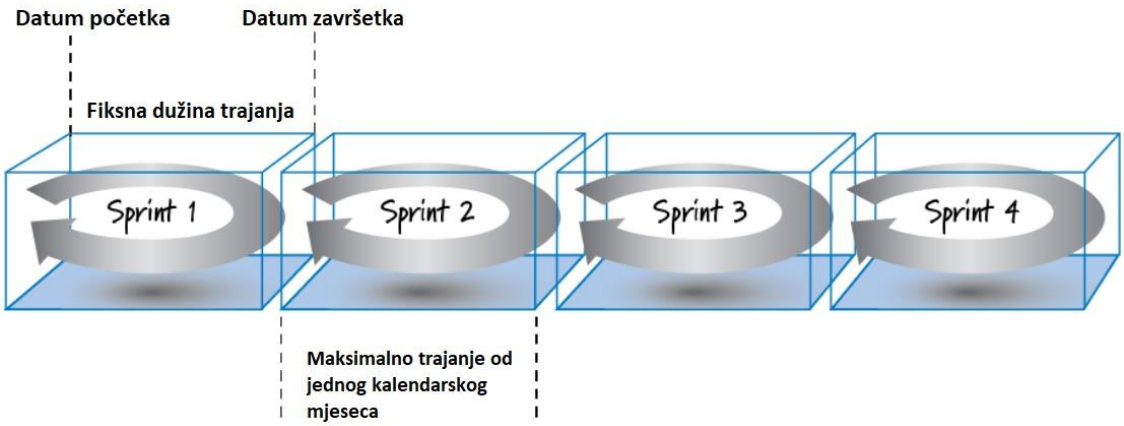
Navedeni događaji su osmišljeni tako da stvore transparentnost i pregled kao provjeru. Isključivanjem bilo kojeg od njih smanjuje transparenost i gubi se mogućnost za pregled i adaptaciju (Schwaber i Sutherland, 2017).

4.3.2.1. Sprint

Osnovnu komponentu Scram-a čini Sprint. To je postupak prilagođavanja promjenjivim varijablama okoline (zahtjevima, vrijemenu, resursima, znanjem, tehnologiji itd.) (Abrahamsson *et al*., 2002). Sprint je vremenski okvir koji uglavnom traje od 1- 4 sedmice, u kojem tim radi sve ka dostizanju zadatih ciljeva. U tom vremenskom periodu se kreira isporučiv i upotrebljiv inkrement. Sama raspodjela razvojnog procesa na Sprintove minimizira potencijalne rizike od stvaranje nekvalitetnog inkrementa ili možda nedovoljno dobrog za tržište, a sa druge strane potpomaže lakšem prilagođavanju novim zahtjevima ili određenim željenim korekcijama od strane korisnika.

Sprint je omotač za druge Scrum događaje o kojima se ranije spominjalo. Kako su sprintovi vremenski ograničeni, time završetkom jednog otpočinje novi sprint (Slika 16).

Slika 16 Izgled i vremensko razgraničenje Sprint-a



Izvor: Rubin (2013)

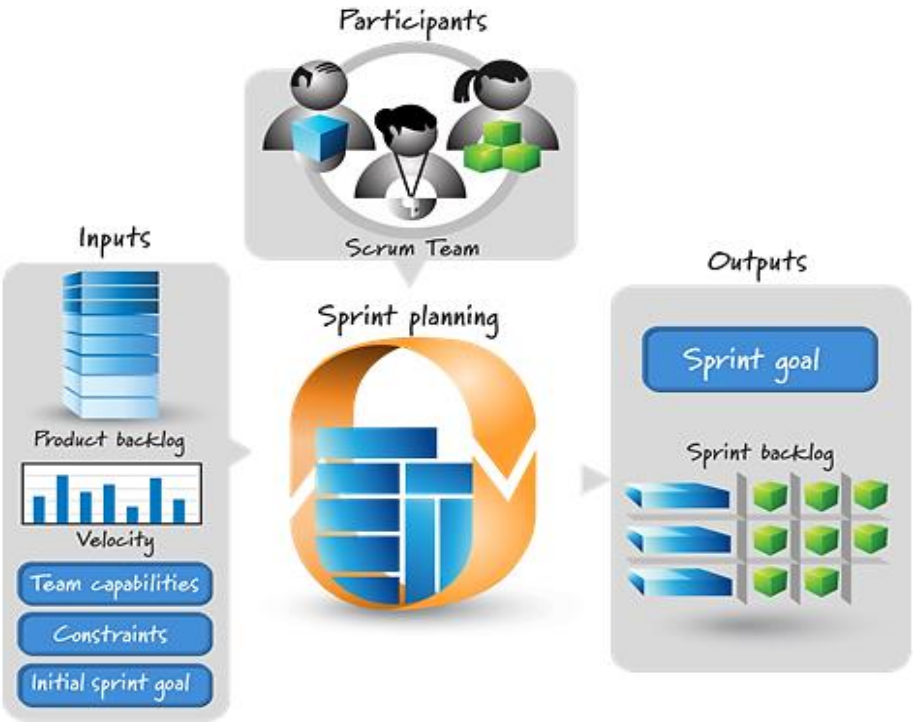
4.3.2.2. Sprint planiranje

Sve aktivnosti koje se trebaju završiti u sprintu se planiraju u ovoj događaju. Sve scrum uloge su uključene u ovaj proces, s tim da sastanak organizuje Scrum Master. On je taj koji podsjeća na svrhu Scrum planiranja i na njegovu vremensku dužinu (Schwaber i Sutherland, 2017).

Trajanje ovog događaja je najviše 8 sati za jednomjesečni sprint, ali to ovisi o dužini sprinta, te zbog toga može biti i kraće (Schwaber i Sutherland, 2017). Ovo je dvofazni sastanak u kojem se u prvoj fazi sastanka donosi odluka o ciljevima i funkcionalnosti sljedećeg Sprinta. Dok u drugoj fazi sastanka Scrum Master i Scrum tim razrađuju načina kako da inkrement proizvoda implementiraju tokom Sprinta (Abrahamsson *et al*., 2002).

Razvojni tim je taj koji planira aktivnosti rada i dizajnira proizvod kako bi se dobio isporučivi inkrement. Rad koji se treba završiti tokom početkom Sprinta se uglavnom dijele na manje zadatke, a sam razvojni tim je taj koji odlučuje na koji način će implementirati Sprint Backlog funkcionalnosti (Schwaber i Sutherland, 2017).

Slika 17 Sprint planiranje



Izvor: Rubin (2013)

4.3.2.3. Dnevni Scrum

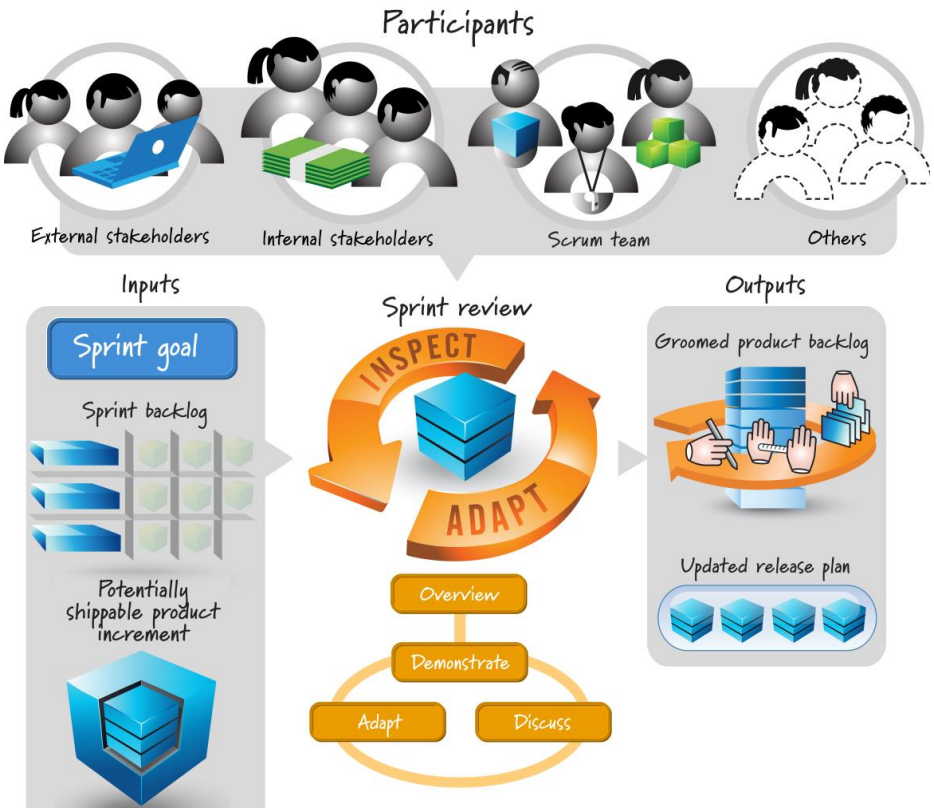
Scrum sadrži svakodnevne stand-up 15 minutne sastanke koji se organiziraju u cilju kontinuiranog praćenja napretka Scrum tima. Ovim sastancima se povećava

transparentnost i uvid u tok rada, čime se povećava produktivnost i efikasnost tima, te se rješavaju potencijalne nedoumice i prepreke. Pored toga, ovim se poboljšava komunikacija, unapređuje se znanje i brže se dolazi do ispunjenja cilja (Schwaber i Sutherland, 2017). Tokom sastanka se odgovara na tri ključna pitanja u kojima se obrazlaže ono što smo do sada uradili, šta će se danas raditi, te koje su nam prepreke za naredni korak u radu. Svi nedostaci ili nejasnoće u procesu razvoja inkrementa se pokušavaju prepoznati i uklanjaju radi poboljšanja procesa. Scrum majstor vodi dnevne sastanke, a Scrum tim je taj koji odgovara na pitanja (Abrahamsson *et al*., 2002).

4.3.2.4. Pregled Sprinta

Posljednjih dana Sprinta organizuje se prezentacija rezultata inkrementa od strane Scrum tima i Scrum Mastera. U ovom neformalnom sastanku učestvuju i eksterni i interni sudionici, korisnici i vlasnik proizvoda. Vremensko trajanje ovog događaja je 4 sata u četvero-sedmičnom sprintu. Cilj je pokretanje diskusije, komunikacije, evaluacije, saradnje u svrhu dobijanja povratnih informacija radi potencijalnih poboljšanja (Schwaber i Sutherland, 2017). Učesnici procjenjuju inkrement, te razmatraju buduće aktivnosti. Ovdje se mogu iznijeti potpuno nove funkcionalnosti Backlog Proizvoda, te se također po potrebi može i promijeniti smjer izgradnje softvera u narednom Sprintu (Abrahamsson *et al*., 2002).

Slika 187 Pregled Sprinta



Izvor: Rubin (2013)

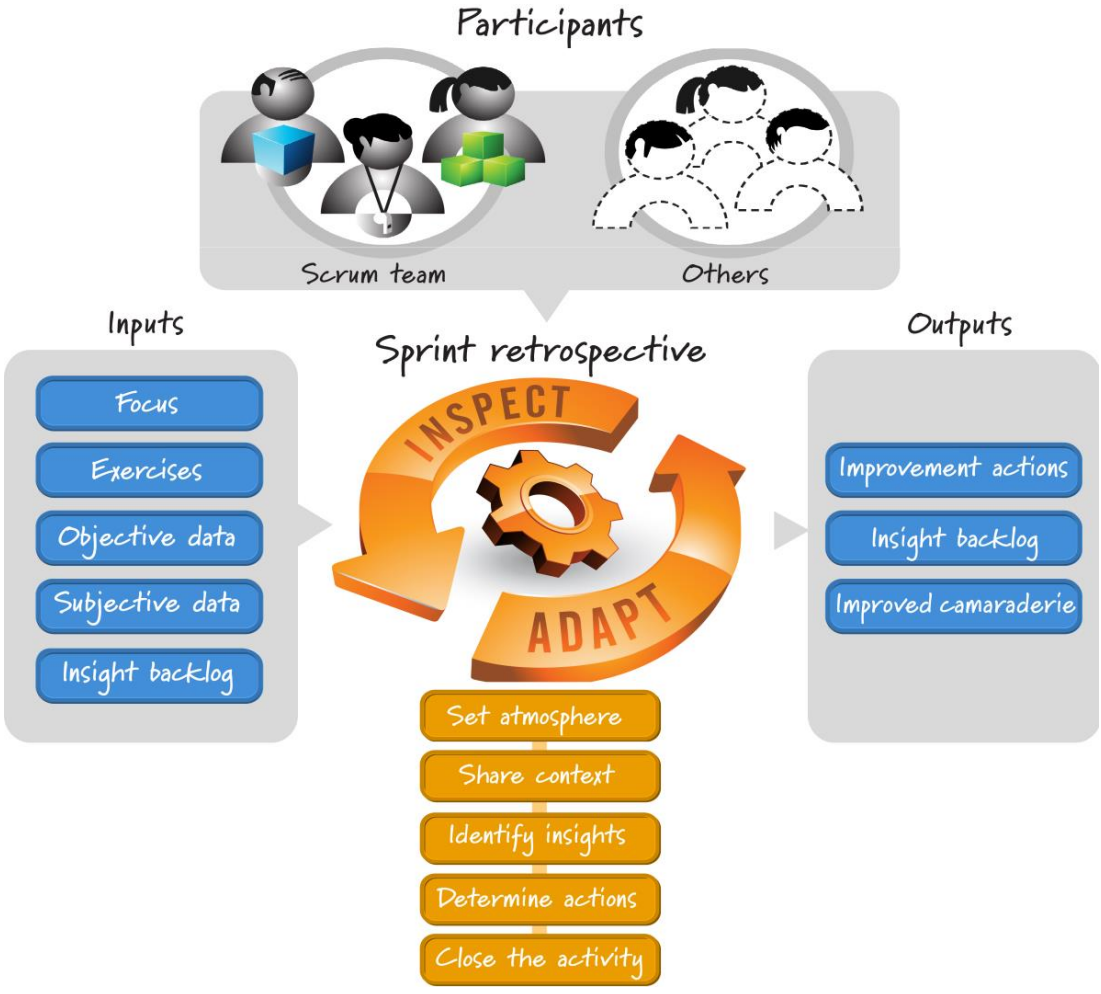
4.3.2.5. Retrospektiva Sprinta

Posljednji događaj Sprinta je retrospektiva u kojoj cijeli tima Scrum-a vrši lični pregled i uvid u ono što su uradili, te se uz to stvara plan za poboljšanje realizirano u novom Sprintu. Putem ovog susreta se prikupljaju objektivni i subjektivni dojmovi, osjećanja u vezi projekta, te se prilažu nove ideje i otvara se prostor za nadogradnju novog znanja kroz dodatne vježbe (Schwaber i Sutherland, 2017).

Scrum master je zadužen za održavanje ovog sastanka koji obično traje 3 sata. Važnost postojanja ovog događaja jeste zbog zadnje provjere Sprinta, analize i poređenja stavki kako su urađene i kako bi se mogle poboljšati, zbog budućih planova unapređenja odnosa, procesa ili alata, te zbog unaprijeđenja interne efikasnosti koja će doprinijeti većoj kvaliteti softvera koji se izdaje (Schwaber i Sutherland, 2017).

Slika 18. ilustruje retrospektivu sprinta.

Slika 198 Retrospektiva Sprinta



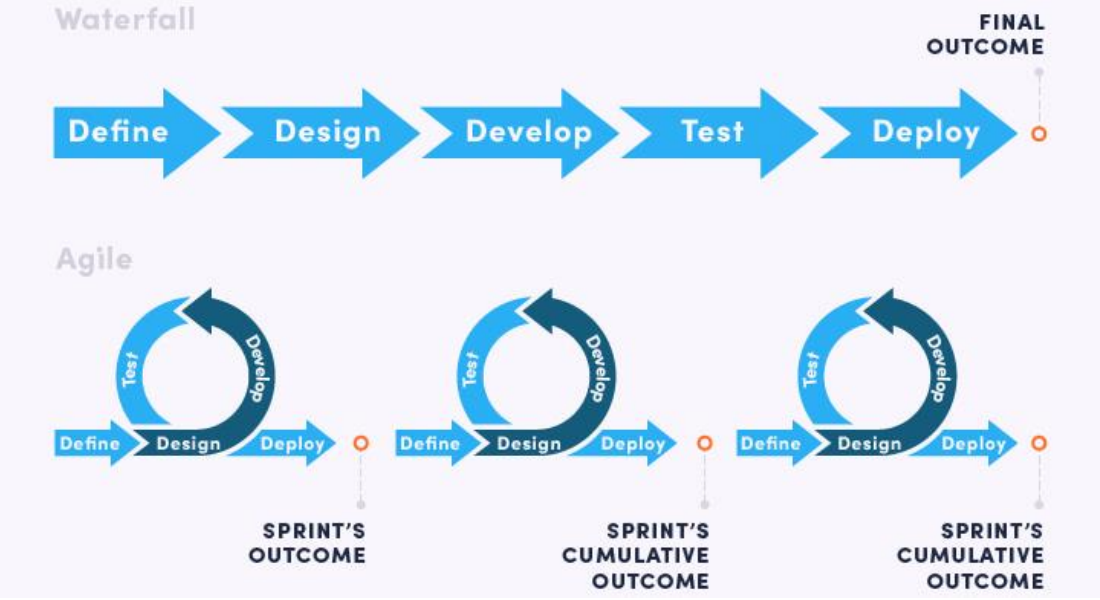
Izvor: Rubin (2013)

5. KOMPARACIJA TRADICIONALNOG I AGILNOG PRISTUPA

Kada govorimo o razlikama između tradicionalne i agilne metodologije, tabele u nastavku teksta ističu jasne naznake njihovih različitosti po određenim segmentima ili parametrima. U tim istim razlikama se zapravo nalaze prednosti ili mane jedne metodologije u odnosu na drugu. Iako tradicionalni pristup upravljanju projektima ističe robusnost kao jednu od svojih prednosti, sa težnjom da se iste metode i tehnike primjenjuju na sve projekte jednolično, zapravo, to može biti jedan od ključnih nedostataka takvog pristupa. Danas sve veći broj autora naglašava činjenicu da „jedna veličina ne pristaje svima“. Projekti, isto kao i poslovna okruženja općenito, postaju progresivno složeni, s većim brojem zadataka, složenim međusobnim odnosima i visokim stepenom težnje za inovativnošću (Špundak, 2014).

Pretpostavka da je projekt izoliran od okoline kosi se sa adaptacijom na brzomijenjajuće trendove i tržišne kompetetivne težnje. Promjena u bilo kojem obliku je stvarnost današnjeg poslovnog okruženja. Određene izmjene u početnom planu su neizbježne zbog adaptacije nepredvidivim i dinamičkim okolnostima u projektnom okruženju ili unutar samog projekta. Također, ponekad je vrlo teško stvoriti cjelovit projektni plan na samome početku zbog nemogućnosti jasno definisanih ciljeva. Kako bi se povećalo vrijeme posvećeno planovima i kontrolama, generira se nesrazmjeran napor u upravljanju koristima u samoj izvedbi projekta (Špundak, 2014). Slika ispod prikazuje vremenske odrednice i komparativna postignuća obje metode, gdje je vremenski okvir trajanja projekta primjenom tradicionalne metode zapravo šansa za čak više kreiranih proizvoda agilnim pristupom.

Slika 20 Poređenje tradicionalne i agilne metodologije



Izvor: Pawlicka (2021)

U nastavku su tabelarno prikazana razlike između tradicionalne i agilne metodologije.

Tabela 3 Razlike između tradicionalne i agilne metodologije

Pristup	Tradicionalni	Agilni
Ciljevi projekta	Fokus na kompletiranje projekta kroz zahtjeve vremena, troškova i kvalitete	Fokusi na poslovne rezultate, postizanje više kriterija uspjeha
Plan projekta	Linearan	Kompleksan, iterativan
Planiranje	Održava se jednom na početku projekta	Izvodi se na početku i održava se kad god je to potrebno
Zahtjevi	Jasni početni zahtevi; niska stopa promjene	Kreativni, inovativni; zahtjevi nejasni
Menadžerski pristup	Krut, sa fokusom na početni plan	Fleksibilna, prilagodljiva varijabla
Rad / Izvođenje	Predvidljivo, mjerljivo, linearno, jednostavno	Nepredvidljiv i nemjerljiv, nelinearan, složen
Organizacijski utjecaj	Minimalan, nepristrasan, od početka projekta	Utiče na projekat tokom njegovog izvođenja
Članovi tima	Nije naglašeno; očekivane fluktuacije; distribuirani tim	Kolocirani tim; manji tim
Kontrola projekta	Identifikuje odstupanja od izvornog plana i ispravlja rad da bi se slijedio	Identifikuje promjene u okruženju i prilagođava plan u skladu s tim
Primjena metodologije	Opća i jednaka primjena u svim projektima	Procesno prilagodljiva ovisno o vrsti projekta
Stil upravljanja	Model služi svim vrstama projekata	Adaptivni pristup, jedan model ne prisustvuje svim vrstama projekata

Izvor: Prilagođeno iz Špundak (2014), Shenhar i Dvir (2007).

Sa gledišta razvoja softvera, Hoda, Noble i Marshall (2010) predstavljaju poređenje između ova dva pristupa u narednoj tabeli.

U nastavku su prikazane razlike između agilnog i tradicionalnog upravljanja projektima softvera.

Tabela 4 Razlike između agilne i tradicionalne metode u IT projektima

Pristup	**Tradicionalni**	**Agilni**
Razvojni model	Tradicionalni	Iterativni
Fokus	Proces	Ljudi
Menadžment	Kontrolira	Olakšava
Uključenost kupaca	Tokom definisanja zahtjeva i u fazi isporuke	Uvijek uključeni
Developeri	Rade pojedinačno unutar timova	U saradnji ili u paru
Tehnologija	Bilo koja	Uglavnom objektno orijentisana
Karakteristike proizvoda	Sve uključene	Prvo najvažnije
Testovi	Na kraju razvojnog ciklusa	Iterativni
Dokumentacija	Potpuna	Samo kad je potrebna
Planiranje	Dugoročno	Kratkoročno
Vrijeme između specifikacije i implementacije	Dugo	Kratko
Vrijeme potrebno za otkrivanje problema	Dugo	Kratko
Rizici vezani za vrijeme implementacije projekta	Visoki	Niski
Odgovor na promjene	Loš	Dobar

Izvor: Prilagođeno iz Wysocki (2014), Hoda, Noble i Marshall (2010).

Pored tabela koja su stvorile temeljni uvid, u nastavku će ukratko istaknute neke od prednosti i mana ove dvije metodologije.

Prednosti Agilnog pristupa:

- Akcenat na saradnji i na aktivnoj uključenosti kupaca u diskusiji o zahtjevima proizvoda čini da programeri razumiju specifičnosti potrebnog softvera što rezultira bržom isporukom proizvoda, bolje prilagođen stvarnosti kupca i s očekivanom kvalitetom (Pereira, Torreão i Maçal**,** 2007; Shiklo, 2017),
- Mogućnost izmjene zahtijeva u bilo kojoj fazi omogućava prilagođavanje softvera potrebama korisnika (Shiklo, 2017),
- Stvara pogodno okruženje za inovacije,
- Smanjeno vrijeme za prikupljanje zahtjeva, smanjenje troškova i rasporeda, te rana isporuka prvog radnog softvera kroz iteraciju (Shiklo, 2017),
- Programeri pristupaju projektu čim postanu svjesni njegovih osnovnih karakteristika, te postepeno uvode druge karakteristike u narednim iteracijama čime se minimiziraju rizici (Azanha *et al*., 2017; Shiklo, 2017),
- Ne teži se temeljnoj dokumentaciji, jer obim projekta se može promijeniti (Shiklo, 2017),
- Testiranje softvera nakon svake iteracije smanjuje akumulaciju grešaka do kraja projekta (Shiklo, 2017),
- Povećana fleksibilnost, transparentnost i produktivnost tima kroz sastanke kroz koje se dobija uvid u projekat (Rahimian i Ramsin, 2008),
- Povratne informacije kupaca su uključene i poboljšavaju isporučeno rješenje (Hillaire, 2018),
- Sposobnost rješavanja nepredvidivosti oslanjajući se na ljude u procesima, daje članovima tima veću diskreciju i ovlasti donošenja odluka kako bi se mogli samoorganizirati i bolje reagirati u novonastalim situacijama, te promiče aktivnosti promišljanja i razmišljanja tako da tim može kontinuirano učiti i prilagođavati se (Nerur *et al*., 2005),
- Naglašava timski rad (Rosenberg i Stephens, 2003), usmjeravajući suradnju između programera i drugih sudionika u samoorganiziranim, višefunkcionalnim timovima (Hoda *et al*., 2013), sa kolektivnim vlasništvom i kolektivnom odgovornošću kao ključnim atributima (Robinson i Sharp, 2005).
- Olakšavanje upravljanja projektom, budući da postoji veća integracija i predanost projektnog tima, koji se posljedično osjeća motiviranijim: moral tima se podiže.

Nedostaci agilnog pristupa:

- Zbog mogućnosti mijenjanja obima projekta, uvijek postoji šansa da naiđete na neke neočekivane izazove i rizike sa kojima se treba nositi (Shiklo, 2017),

- Rukovodioci projekata se takođe suočavaju sa izazovima, poput oslobađanja autoriteta koji dolazi sa stilom komande i kontrole i prihvatanja njihove uloge facilitatora/koordinatora u agilnom pristupu (Nerur *et al*., 2005),
- Novi izazovi postavljaju se kada timovi moraju raditi s kraćim životnim ciklusima (Boehm i Turner, 2005),
- Izazovi u poslovnim procesima nastaju kada organizacijska kultura zahtijeva precizna predviđanja zadataka i ne potiče eksperimentisanje ili evoluciju (Boehm i Turner, 2005).

Prednosti vodopad modela:

- Dobar je za projekte kod kojih su potpuno definisani i jasni zahtjevi i koji ne zahtijevaju visok nivo inovativnosti,
- Podržava stav „definiši prije dizajna“ i „dizajniraj prije implementacije“,
- Unaprijed poznata procjena i kontrola svih poslova potrebnih za dovršetak projekta (vrijeme, trošak, resurs) (Hillaire, 2018),
- Pažljivo i opsežno planiranje može doprinijeti da se odma identifikuju potencijalni rizici i da se pronađu rješenja za njih (Nerur *et al*., 2005),
- Svaka faza ima specifične izlazne proizvode i proces analize.

Nedostaci vodopad modela:

- Nefleksibilnost prema promjenama (Hillaire, 2018),
- Koncentrira se na detaljno planiranje, što potiče uski, operativni pogled na projekt (Hillaire, 2018),
- Nepredvidiva kvaliteta softvera zbog kašnjenja testiranja do kraja projekta (Petersen *et al*., 2009),
- Prikupljanje zahtjeva u fazi analize kada se stvara previše dokumentacije koja se nikada kasnije ne koristi u projektu (Petersen *et al*., 2009),
- Promjena obima poslova u toku životnog ciklusa projekta može uništiti projekat,
- Većinom se fokusira na pristupe na taktičkom nivou, ali su pristupi na strateškom nivou neophodni zbog sve složenije i kritičnije prirode projekata (Hillaire, 2018),
- Provjera koja se događa u fazi testiranja može biti izazovna ako prekoračenje rasporeda ograniči vrijeme testiranja preostalo na kraju projekta (Petersen *et al*., 2009),
- Ne podstiče na učestale sastanke i povratne informacije od strane klijenta,
- Visok stepen neizvijesnosti o finalnom proizvodu za čiji se razvoj jako duge čeka,
- Ne postoji konkretna radna verzija softvera dok se cijeli projekat ne kompletira,
- Nije pogodan za složene i objektno-orijentisane projekte,
- Previše utrošenog vremena za planiranje i manjak vremena za izradu,
- Nemogućnost na samom početku procijeniti sve potrebe klijenta,
- Povećani rizici od grešaka i neuspjeha projekta.

6. ISTRAŽIVANJE I ANALIZA REZULTATA

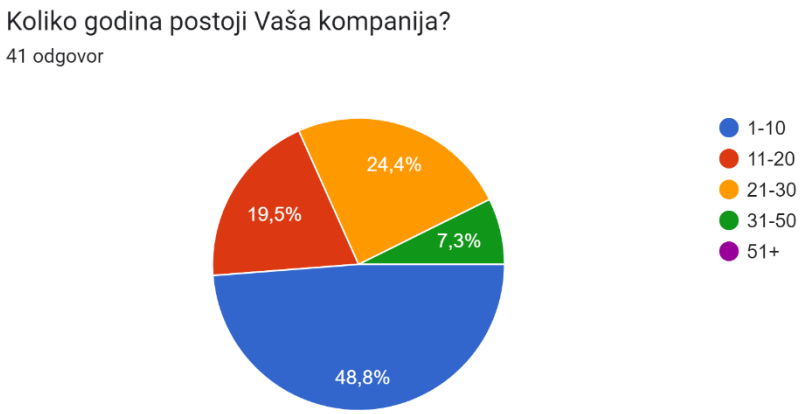
U ovom istraživanju za uzorak su uzete IT kompanije sa područja Bosne i Hercegovine, kako bi se potvrdili ciljevi istraživanja i ispitale zadane hipoteze. Anketiranje je sprovedeno u oktobru mjesecu 2023. godine i u njemu su učestvovali 41 predstavnik IT kompanija rasprostranjenih na području Sarajeva, Tuzle, Mostara, Banja Luke i Bihaća. Popis kompanija nalazi se u Prilogu br. 1.

Osnovni cilj ankete bio je utvrditi da li agilna metoda ima veći postotak primjene u IT industriji u BiH u odnosu na tradicionalnu, da li su njeni temeljni principi razlogom njene veće primjene i ustanoviti da li postotak primjene Scrum-a ukazuje na njegovu veću primjenu u odnosu na druge agilne podmetodologije.

Anketa je obuhvatala 21 pitanje, koje je navedeno u Prilogu br. 2. Prvo se ispitala struktura uzorka po godini poslovanja i brojem uposlenika, te opće informacije o ispitaniku koji je davao odgovore. Potom, biranjem između agilne i tradicionalne metodologije, svaki učesnik je imao mogućnost da dalje odgovara u dva ponuđena smjera. Oni koji su izabrali agilnu metodologiju dobili su jedan set pitanja, a oni koji su izabrali tradicionalnu, dobili su drugi set pitanja.

6.1. Rezultati strukture uzorka

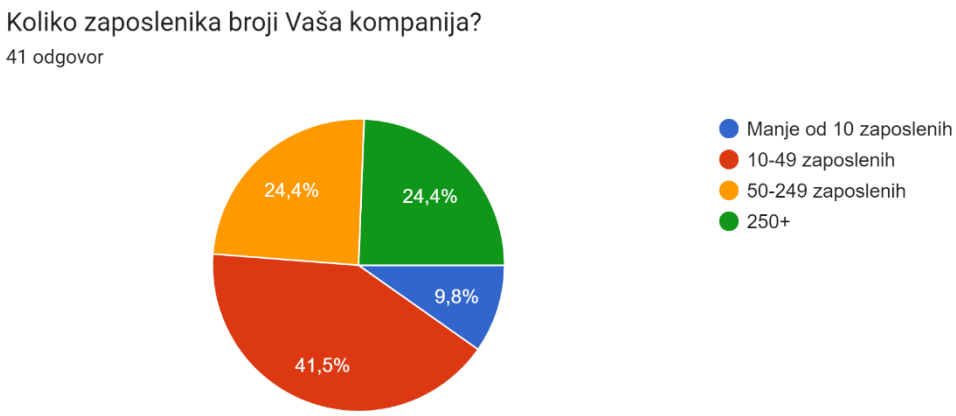
Graf 1 Godine postojanja kompanije



Izvor: Autor završnog rada

Prema godinama postojanja, najveći postotak (48,8%) obuhvataju kompanije koje postoje između 1-10 godina, što ukazuje da su većinski mlade kompanije i da se ovaj tip industrije jako širio posljednjih godina.

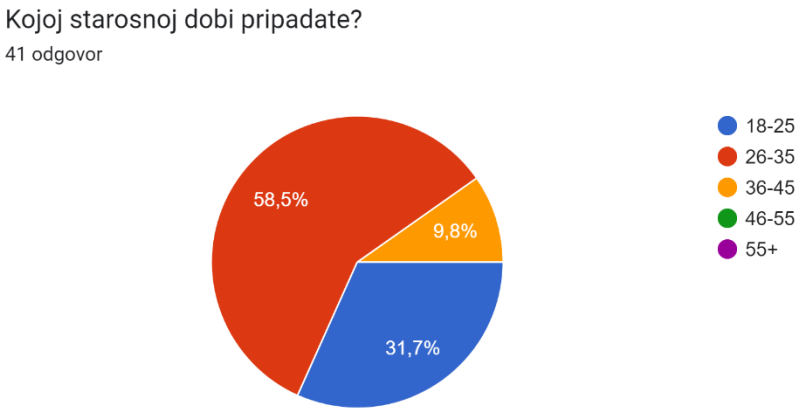
Graf 2 Broj uposlenika



Izvor: Autor završnog rada

Kada govorimo o broju uposlenika, najveći udio nose kompanije koje imaju između 10-49 zaposlenih i taj procenat iznosi 41.5%, što zapravo predstavlja da su u pitanju mala preduzeća. Ujednačen udio nose kompanije koje broje od 50-249 i kompanije sa 250+ uposlenika, koje predstavljaju srednja i velika preduzeća. Najmanji udio nose kompanije sa manje od 10 uposlenika, a to je 9,8%. Ovim jasno vidimo da IT kompanije pored naglog širenja posljednjih godina, imamo i njihov unutrašnjih razvoj povećanjem brojem uposlenih, gdje umjesto mikro preduzeća, najveći procenat iznose mala preduzeća.

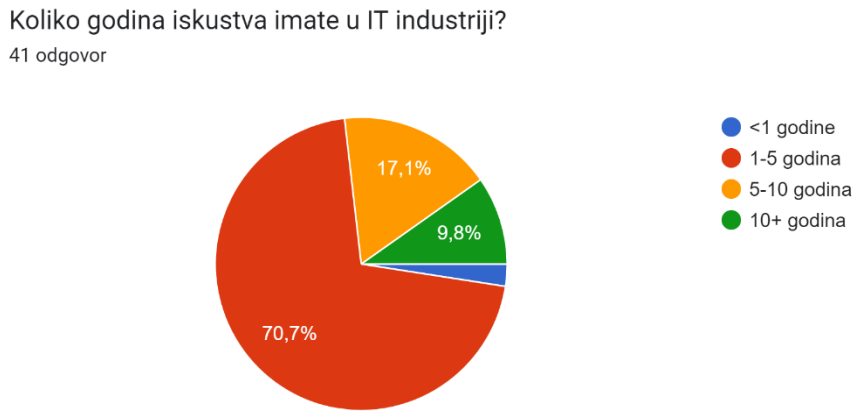
Graf 3 Starosna dob ispitanika



Izvor: Autor završnog rada

Ovoj anketi su najviše pristupili učesnici starosne dobi između 26-35 godina, gdje ih je u procentima bilo 58,5%. Potom učesnici između 18-25 godina, u procentu od 31,7% i naposljetku osobe između 36-45 godina. Ove dvije skupine pripadaju generaciji Milenijalci i Generacija Z, što možemo uvrstiti sve u mlađu populaciju ljudi.

Graf 4 Godine iskustva



Izvor: Autor završnog rada

Najviše su uzeli učešće učesnici sa 1-5 godina iskustva u IT industriji. Same godine iskustva i mladost učesnika će između ostalog u nastavku pokazati koliko su oni bili spremni da naprave iskorak u primjenu novijih projektnih metodologija.

6.2. Rezultati odabira metodologije

Graf 5 Procenat primjene tradicionalne i agilne metode

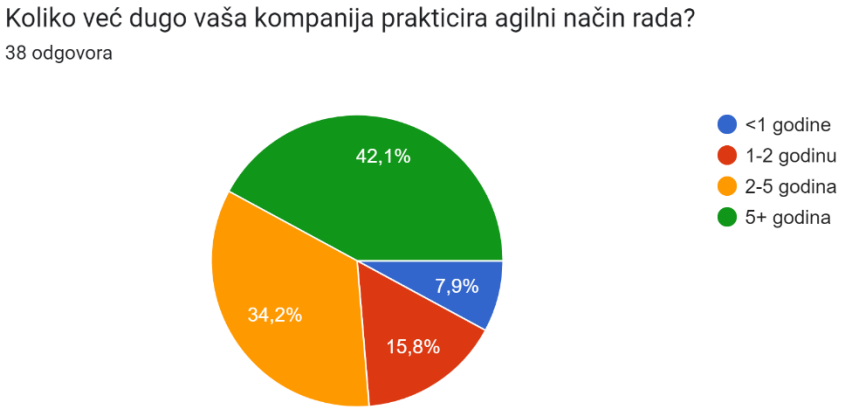
Izvor: Autor završnog rada

Prva hipoteza koja se trebala ispitat ovim istraživanjem, a čiji odgovor nalazimo u grafu iznad je: **H1:** Agilne metode projektnog menadžmenta se više primjenjuju u IT kompanijama u BiH u odnosu na tradicionalne.

38 ispitanika od ukupnih 41, a to je 92,7%, je odgovorilo da većinski primjenjuju **agilnu metodu** projektnog menadžmenta, a sa druge strane, samo **3 ispitanika**, što iznosi 7,3% odgovorilo je da koriste **tradicionalnu**. Ovdje vidimo izrazito visok procenat razlike i vodstva agilne metodologije u odnosu na tradicionalnu, čime se **prihvata zadana hipoteza**.

6.3. Rezultati iz oblasti agilne metodologije

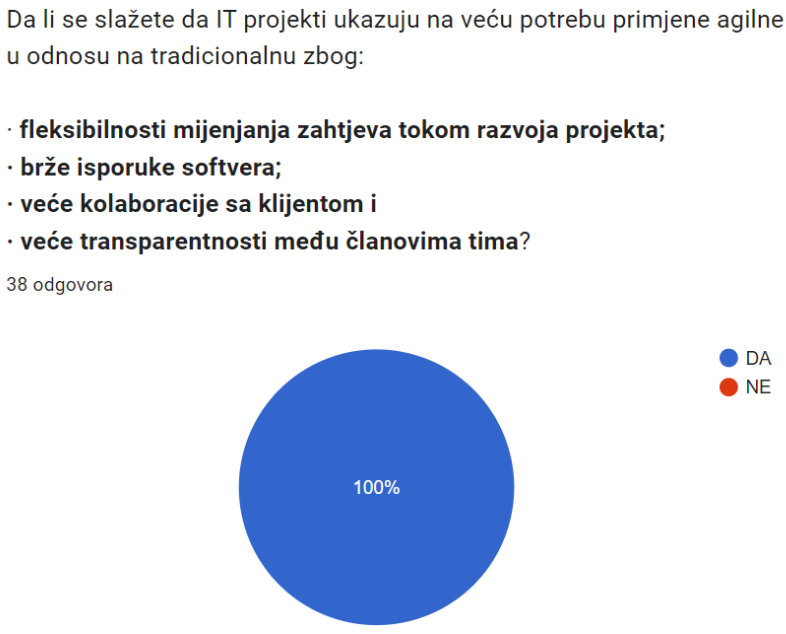
Graf 6 Period praktikovanja agilnog načina rada



Izvor: Autor završnog rada

Već se ranije spominjalo da su u ovoj anketi najviše učestvovala mala preduzeća. Najviše odgovora na godine praktikovanja agilnog načina rada i to 5 i više godina je iznosilo 42,1 %. Potom, imamo 2-5 godina primjene agilnog načina rada u procentima od 34,2%. Ovo je i više nego pozitivan znak da su se kompanije već dugo ili čak od samog osnivanja odlučile na agilni način rada i da su u njemu prepoznale najadekvatniji pristup poslovnim projektima.

Graf 7 Agilni principi i primjena agilne metode



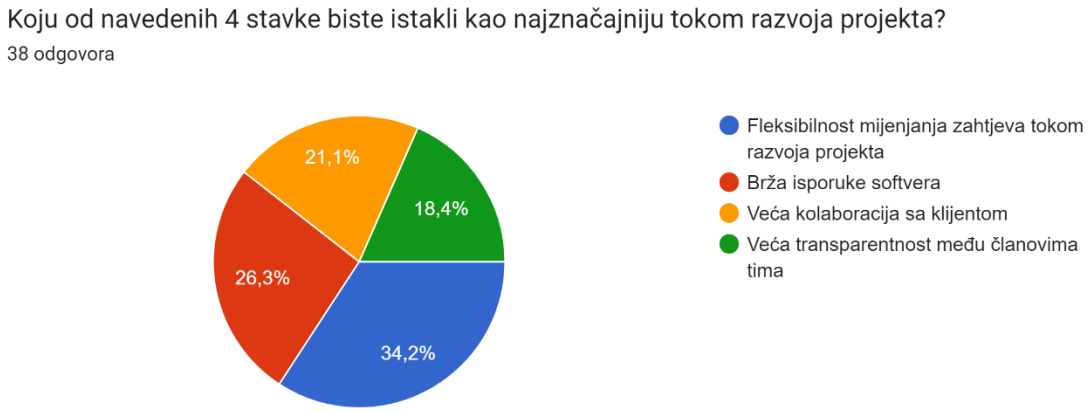
Izvor: Autor završnog rada

Kako bismo ustanovili da li su temeljni agilni principi glavni razlozi korištenja agilne metode umjesto tradicionalne, ili su to ipak neki drugi, bila je postavljena sljedeća hipoteza:

H2: IT projekti ukazuju na veću potrebu primjene agilne metodologije u odnosu na tradicionalnu zbog fleksibilnosti mijenjanja zahtjeva tokom razvoja projekta; brže isporuke softvera; veće kolaboracije sa klijentom i veće transparentnosti među članovima tima.

U ovom slučaju svi ispitanici su odgovori jednoglasno „DA“ što je iznosilo 100%, te time potvrdili da ono na čemu je agilna metodologija primarno zasnovana su bili dovoljno jaki razlozi u odnosu na sve potencijalno drugačije, da se kompanije odluče za agilni način rada. Ovim se **zadata hipoteza potvrđuje i prihvata.**

Graf 8 Najistaknutija stavka agilne metode



Izvor: Autor završnog rada

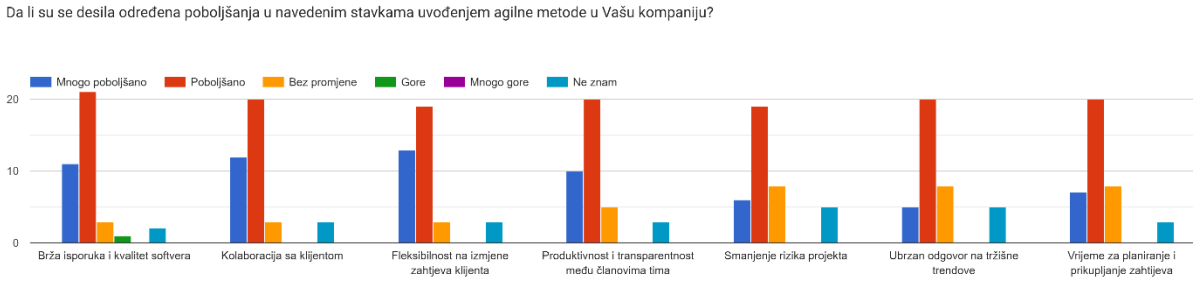
U prikazanom grafu su odgovori poprilično ujednačeni, sa blagom prednošću „Fleksibilnosti mijenjanja zahtjeva tokom razvoja projekta“, čime vidimo da ipak taj segment mogućnosti pravljena potencijalnih izmjena tokom projekta je izuzetno značajan za IT firme.

Graf 9 Prednosti agilne metodologije

Izvor: Autor završnog rada

U ovom grafu možemo da vidimo da su ispitanici za najistaknuti prednosti agilne metodologije izglasali najviše „Smanjenje broja grešaka i problema u projektu“ i „ Smanjenje rizika i bolja kontrola nad projektom“. Samim tim što je tradicionalni model više rigidan, stvara pogodno tlo propadanje istog shodno nemogućnosti da se odupre promjenama. Mogućnost izmjene zahtjeva uz kraće iteracije, veća suradnja sa klijentom i bolja organizovanost tima otvara vrata mogućnosti i smanjenja grešaka i smanjenja rizika projekta.

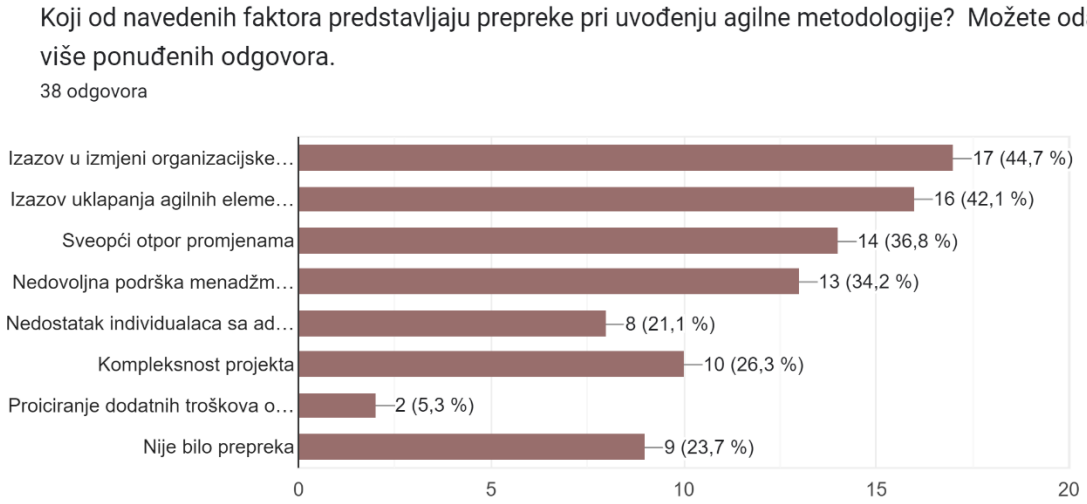
Graf 10 Poboljšanja pri uvođenju agilne metode



Izvor: Autor završnog rada

Prema grafu stepena poboljšanja određenih stavki uvođenjem agilne metode, učesnici su najviše biralni opciju da su se desila poboljšanja, gdje haman da nemamo nigdje negativne rezultate.

Graf 11 Prepreke pri uvođenju agilne metode



Izvor: Autor završnog rada

Kada se govori o preprekama pri uvođenju agilne metode, najviše su uvrštene prepreke kao što su: izazov u izmjeni organizacijske kulture, izazov uklapanja agilnih elemenata u prethodni ustaljeni tradicionalni okvir, te sveopći otpor promjenama. Ovo sve implicira da ipak za uvođenje određene metodologije mora postojati organizacijska kultura spremna na

promjene i novitete, te da mora postojati način na koji bi se najlakše adaptirali elementi prethodne metodologije u noviju i drugačiju.

Graf 12 Budući planovi primjene agilne metode

Izvor: Autor završnog rada

Za buduće projekte je 36 ispitanika izglasalo da planira koristiti agilni način rada, dok je dvoje izglasalo opciju da nisu sigurni.

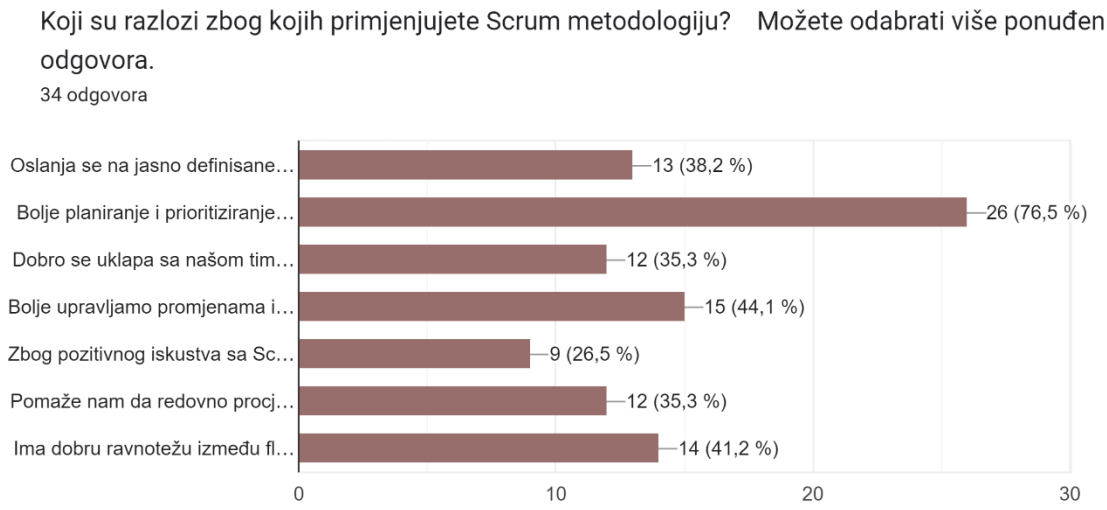
Graf 13 Stepen zastupljenosti agilnih metodologija

Izvor: Autor završnog rada

Kada su ispitanici agilne metodologije dobili izbor da izdvoje najzastupljeniju agilnu metodologiju, njihovi odgovori su bili sljedeći: 34 učesnika (89,5%) je označilo da radi po Scrum metodologiji, dok su po dva učesnika (5,3%) izabrala da rade po „Feature driven“ razvojnom okviru i po Kanbanu. Iz priloženog se može vidjeti da su i druge metodologije zastupljene mimo Scrum-a, ali u jako malom procentu u odnosu na Scrum koji je ubjedljivo **potvrdio treću hipotezu** i ona se u ovom slučaju **prihvata**.

H3: SCRUM metodologija se najviše koristi u IT projektnom menadžmentu u BiH u odnosu na sve ostale agilne metodologije.

Graf 14 Razlozi primjene Scrum-a

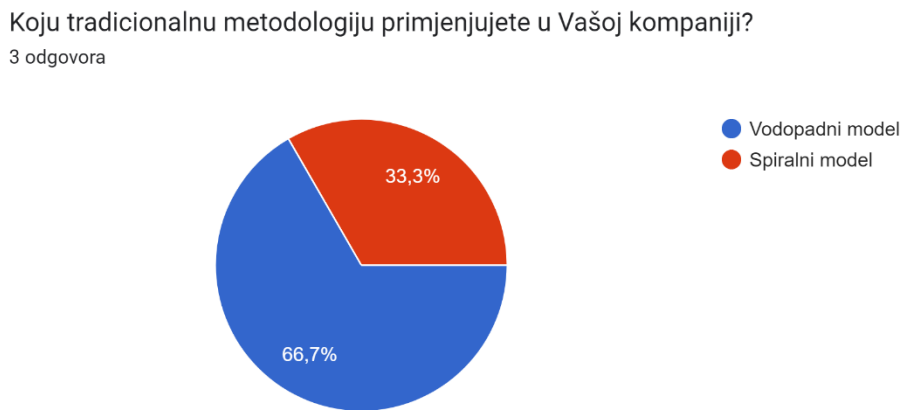


Izvor: Autor završnog rada

Kada se govori o razlozima odabira Scrum-a, bolje planiranje i prioritiziranje zadataka kroz Product Backlog i Sprint Backlog se izdvojilo po broju glasova, gdje je njih 26 odabralo ovaj faktor pozitivnog uticaja, a potom odgovor sa 15 glasova je bolje upravljanje promjenama i nepredvidivim situacijama. Menadžment, dobra organizacija je očigledno potrebna IT kompanijama uz fleksibilnost na moguće izmjene, jer one same kao takve ne ovise samo o kvaliteti koda i kreiranog softvera, već i cijelom jednom projeknom procesu koji uključuje različite resurse.

6.4. Rezultati iz oblasti tradicionalne metodologije

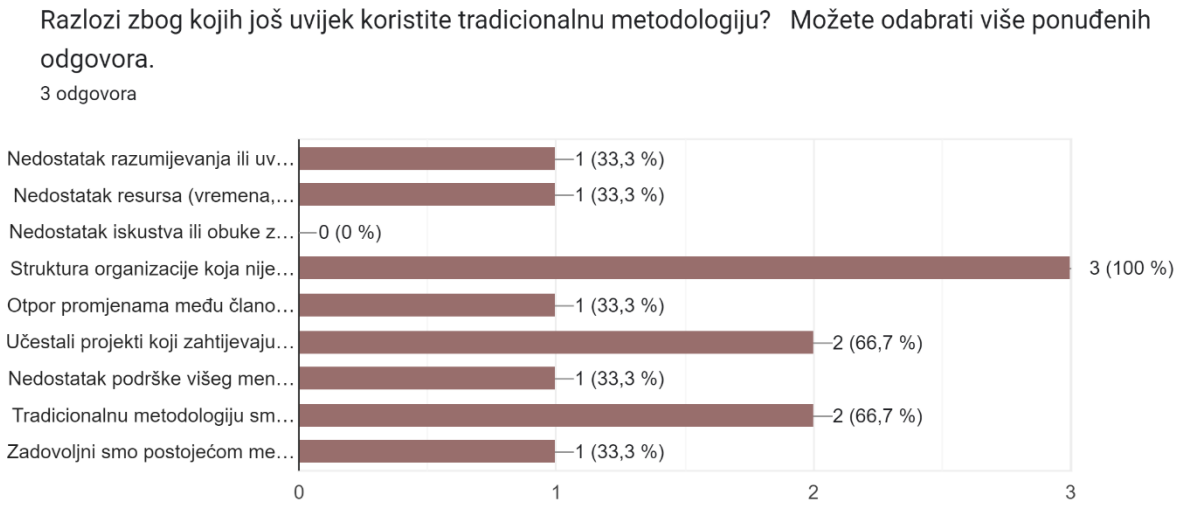
Graf 15 Odabir tradicionalne metodologije



Izvor: Autor završnog rada

Kao što je već ranije rečeno, samo tri ispitanika su glasali za tradicionalnu metodologiju, gdje su njih dvoje izabrali vodopadni model, a jedan ispitanik spiralni. U ovom slučaju veći procenat nosi vodopad model.

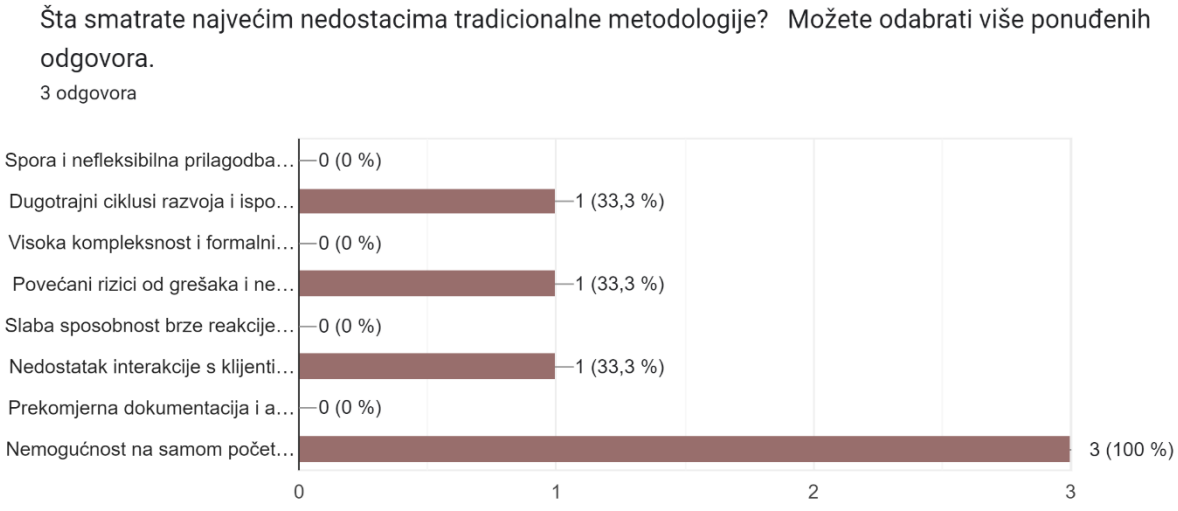
Graf 16 Razlozi korištenja tradicionalne metodologije



Izvor: Autor završnog rada

Najveći razlog zbog kojeg još uvijek koriste tradicionalnu metodologiju po broju glasova je „Struktura organizacije koja nije kompatibilna s agilnom metodologijom“.

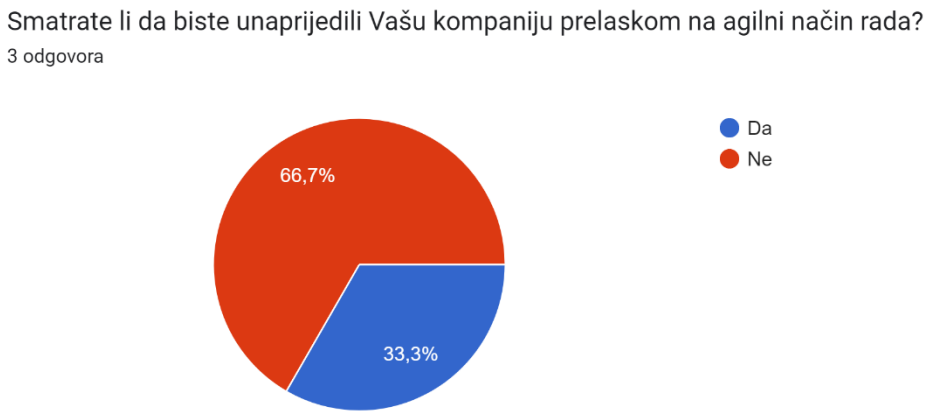
Graf 17 Nedostaci tradicionalne metodologije



Izvor: Autor završnog rada

Kao najveći nedostatak tradicionalne metodologije izdvojen je „Nemogućnost na samom početku procijeniti sve potrebe klijenta“, što je u suštini jedna i od najvećih slabosti tradicionalne metodologije.

Graf 18 Potencijalni prelazak na agilnu metodologiju



Izvor: Autor završnog rada

Na pitanje da li bi unaprijedili svoju kompaniju prelaskom na agilni način rada, dvoje je stavilo „Ne“, a jedno „Da“. Ovim vidimo da nekim IT firmama iz određenih okolnosti nije potreban drugačiji način rada shodno njihovim okolnostima, projektima i uslugama koje nude. Mora se uzeti u obzir svakako da sve kompanije ne kreiraju iste proizvode i usluge, te da ne rade sve na principu stvaranja inovativnih rješenja. Neke od njih kreiraju jednostavna i već poznata rješenja koja su kao takva potrebna klijentima kako bi samo koristili svrsi i potrebi isključivo uposlenicima neke firme, ali ne i njihovim klijentima.

Graf 19 Plan prelaska na agilnu metodologiju

Izvor: Autor završnog rada

Ovo pitanje se svakako nadovezuje na prethodno i ovim potvrđujemo da određene kompanije ne vide potrebu za drugačijom metodologijom u odnosu na trenutnu i da se ničim ne iziskivaju potreba za tim. Dva ispitanika su glasali da ne planiraju preći na agilni način rada, dok s druge strane, imamo jednog koji je odabrao da ima u planu to uraditi. Iz priloženog se može vidjeti da svakako nisu svi trenutno na tradicionalnoj metodologiji iz potrebe za njom, već iz prepreke prelaska na agilnu, gdje u svakom slučaju se teži preći na istu.

7. ZAKLJUČAK

Metoda projektnog menadžmenta kao disciplina za efikasno upravljanje resursima i procesima i to sve sa ciljem ostvarivanja željenog rezultata, igra ključnu ulogu u uspješnom vođenju IT projekata u Bosni i Hercegovini. Uočava se da je IT industrija u zemlji u konstantnom rastu, a da bi se iskoristile sve prednosti ovog rasta, neophodno je primijeniti adekvatnu i efikasnu projektnu metodologiju. Agilna metodologija se ističe svojom fleksibilnošću i naglaskom na iterativnom radu, što je posebno važno u dinamičnom okruženju kao što je IT industrija. Sa druge strane, tradicionalna metodologija se primjenjuje kada su zahtjevi dobro definisani i stabilni.

Svaka od ovih metodologija ima svoje prednosti i nedostatke, međutim važno je razumjeti specifične potrebe projekta, tima i klijenta kako bi se odabrala odgovarajuća strategija. Konkretno u ovom radu se istakao Scrum, kao jedan od najpopularnijih agilnih okvira, kojeg odlikuju prilagodljivost promjenama, kratke iteracije (sprintovi), definisane timske uloge, bliska saradnja i redovna komunikacijama sa klijentom.

Kroz analizu različitih pristupa, kao što su agilni i tradicionalni, urađeno je istraživanje na području Bosne i Hercegovine, u kojem su učestvovali 41 ispitanik iz različitih IT kompanija. Struktura uzorka nam je pokazala da su čak 41,5% ove industrije mala preduzeća koja uglavnom broje između 10-49 uposlenika, čija godina postojanja varira između 1-10 godina. Pored toga što su u pitanju mlade firme, ovoj anketi su pristupili mlađa populacija ispitanika, Milenujalci i Generacije Z, gdje je 70,7% bilo onih sa 1-5 godina iskustva u IT industriji. Da li mlade i male firme, sa mlađom populacijom uposlenika su pokazale spremnost na primjenu i adaptaciju novijih metodologija ili su pobornici tradicionanih, možemo da ustanovimo kroz prezentaciju narednih rezultata.

Istraživanje je dokazalo i prihvatilo prvu hipotezu koja tvrdi da agilne metode projektnog menadžmenta se više primjenjuju u IT kompanijama u BiH u odnosu na tradicionalne. Rezultati su pokazali da se 92,7% koristi agilna metodologija, a 7,3% tradicionalna, što ukazuje na izrazito veliku prednost agilne.

Svi ispitanici koji praktikuju agilni način rada, glasali su da četiri osnovna principa agilnog pristupa su razlogom veće primjene agilne umjesto tradicionalne metodologije, čime se potvrdila i prihvatila druga hipoteza. Ona je glasila: „IT projekti ukazuju na veću potrebu primjene agilne metodologije u odnosu na tradicionalnu zbog fleksibilnosti mijenjanja zahtjeva tokom razvoja projekta; brže isporuke softvera; veće kolaboracije sa klijentom i veće transparentnosti među članovima tima“.

42,1% preduzeća se izjasnilo da već 5 i više godina praktikuju agilni način rada i to ukazuje da su prepoznali određene beneficije iste, s tim da je 94,7% njih glasalo da planira i u budućnosti primjenjivati agilnu metodologiju.

Istaknute prednosti agilne metodologije u odnosu na tradicionalnu koje su izdvojili bile su smanjenje broja grešaka i problema u projektu, te manji rizik i bolja kontrola nad projektom. Samim uvođenjem agilne metodologije najveći procenat firmi su imale poboljšanja u narednim stavkama: brža isporuka i kvalitet softvera, veća kolaboracija sa klijentom, fleksibilnost na izmjene zahtjeva klijenta, produktivnost i transparentnost među članovima tima, smanjenje rizika projekta, ubrzan odgovor na tržišne trendove, te kraće vrijeme za planiranje i prikupljanje zahtijeva. Izazovi u izmjeni organizacijske kulture i uklapanja agilnih elemenata u prethodni ustaljeni tradicionalni okvir su prepoznati kao najveće prepreke pri uvođenju agilnog načina rada u firme.

Ono što je bilo posebno važno ustanoviti, pored saznanja da se agilna metodologija najviše koristi na prostorima Bosne i Hercegovine, jeste koja je njena podmetodologija najrasprostranjenija. Sama treća hipoteza je tvrdila da se SCRUM metodologija najviše koristi u IT projektnom menadžmentu u BiH u odnosu na sve ostale agilne metodologije, a rezultati istraživanja su to i dokazali, čime je ona potvrđena i prihvaćena. Scrum metodologija se koristi u 89,5% kompanija, dok je 5,3% njih glasalo za Kanban i 5,3% za „Feature driven“ razvojni okvir. Bolje planiranje i prioritiziranje zadataka kroz Product Backlog i Sprint Backlog su prema glasanju bili (pored ostalih razloga), vodeći zbog kojeg im je više adekvatan Scrum u odnosu na druge agilne metode.

Kao što je već ranije pomenuto, 7,3% firmi koristi tradicionalni model projektnog menadžmenta gdje dvije od njih koriste vodopadni, a jedna spiralni model. Postoji nekoliko razloga zbog kojih se i dalje opredeljuju za korištenje tradicionalne metodologije u mnogim projektima, a to su: struktura organizacije koja nije kompatibilna s agilnom metodologijom i učestali projekti koji zahtijevaju tradicionalnu metodologiju. Najvećim nedostatkom tradicionalne metode smatraju nemogućnost da na samom početku procijene sve potrebe klijenta. Dva ispitanika su rekla da ne bi unaprijedili kompaniju i da nemaju namjeru preći na agilni način rada, dok je jedan odgovorio da smatra da bi se kompanija unaprijedila i da ima u planu preći.

U svjetlu ovih rezultata, možemo zaključiti da je IT industrija u Bosni i Hercegovini snažno orijentisana ka primjeni agilnih metodologija, najviše Scrum-u, kako bi postigla veću efikasnost i prilagodljivost u dinamičnom okruženju. Agilne metodologije sa svojim principima donose inovaciju razvoja softvera i implementaciju informacionih sistema, poboljšavajući uspješnost projekata, ali istovremeno postavljajući nove složene izazove pred organizacije. Važno je naglasiti da, uprkos popularnosti određene metode, nijedna metodologija nije univerzalno rešenje koje odgovara svim firmama i svim granama industrije. Razlozi mogu biti različiti, ali kroz kontinuirano usavršavanje i primenu najboljih metoda projektnog menadžmenta, IT kompanije mogu ostvariti svoj puni potencijal i doprinjeti svom daljem rastu i razvoju.

REFERENCE

1. Abrahamsson, P., Salo, O., Ronkainen, J. i Warsta, J. (2002) *Agile software development methods: Review and analysis*, VTT publication 478, Espoo, Finland, 107p
2. Agile Alliance (2021). *Kanban* . Dostupno na: https://www.agilealliance.org/glossary/kanban/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'kanban))~searchTerm~'~sort~false~sortDirection~'asc~page~1) (Pristupljeno: 17 Juli 2020)
3. Agile Alliance (2021). *Scrum* . Dostupno na: https://www.agilealliance.org/glossary/scrum/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'scrum))~searchTerm~'~sort~false~sortDirection~'asc~page~1) (Pristupljeno: 25 Decembar 2020)
4. Agile Alliance (2021). *What is Extreme Programming (XP)?*. Dostupno na: https://www.agilealliance.org/glossary/xp/#q=~(infinite~false~filters~(postType~(~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'xp))~searchTerm~'~sort~false~sortDirection~'asc~page~1) (Pristupljeno: 7 Septembar 2020)
5. Arvin, D. (2021). *What is Agile Methodology- Know the What and How?*. Dostupno na: https://www.edureka.co/blog/what-is-agile-methodology/ (Pristupljeno: 7 Septembar 2020)
6. Augustine, S. (2005). *Managing Agile Projects*. Virginia: Prentice Hall Professional Technical Reference.
7. Azanha, A., Argoud, Carmango Junior, J. i Antoniolli, P. (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.
8. Bassil, Y. (2012). A simulation model for the waterfall software development life cycle. *Lebanese Association for Computational Sciences (LACSC), International Journal of Engineering & Technology* (iJET), 2(5), 742-749. Preuzeto sa https://arxiv.org/abs/1205.6904
9. Beck, K. (2000). *Extreme programming explained: Embrace change*. (1st ed.). Addison-Wesley.
10. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Kern, J. (2001). *Manifesto for Agile Software Development*. Dostupno na: http://agilemanifesto.org/ (Pristupljeno: 9 August 2020)
11. Biznis Info (2020). *IT sektor u BiH u usponu: Evo ko je lider!*. BiznisInfo. Dostupno na: https://www.biznisinfo.ba/it-sektor-u-bih-u-usponu-evo-ko-je-lider/ (Pristupljeno: 14 August 2020)
12. Boehm, B. (1988). A spiral model of software development and enhancement. *Computer*, *21*(5), 61-72.

13. Boehm, B. i Turner, R. (2005). *Management challenges to implementing agile processes in traditional development organizations.* IEEE Software, 22(5), 30–39.
14. Cervone, H.F. (2011). *Understanding agile project management methods using Scrum.* OCLC Systems & Services: International Digital Library Perspectives, 27 (1), 18-22.
15. Charvat, J. (2003). *Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processes for Projects.* Hoboken, NJ: John Wiley & Sons, Inc.
16. Ciric, D., Lalic, B., Gracanin, D., Palcic, I., i Zivlak, N. (2018). Agile Project Management in New Product Development and Innovation Processes: Challenges and Benefits Beyond Software Domain. *2018 IEEE International Symposium On Innovation And Entrepreneurship (TEMS-ISIE),* 1-9.
17. Coad, P., LeFebvre, E.¸i De Luca, J. (2000). *Java Modeling In Color With UML: Enterprise* Components and Process, Prentice Hall.
18. Cockburn, A. (1998). *Surviving Object-Oriented Projects: A Manager's Guide,* Addison Wesley Longman.
19. Cockburn, A. (2002) Agile software development joins the "Would-Be" crowd. *Cut. IT J.* 15 (1), 6e12.
20. Cockburn, A. (2002a). Agile Software Development. Boston, Addison-Wesley.
21. Cockburn, A. (2003). *People and Methodologies in Software Development*. Doctoral Dissertation. University of Oslo, Oslo, Norway
22. Cockburn, A. (2006*). Agile Software Development: The Cooperative Game*. Second Edition. Boston, MA: Addison Wesley Professional, Pearson Education, Inc
23. Cohen, D., Lindvall, M., i Costa, P. (2004). An Introduction to Agile Methods. *Advances In Computers, 62,* 1-66.
24. Digite. (2013). *What is Kanban? an overview of the Kanban method*. Preuzeto 14. Augusta 2020, sa https://www.digite.com/kanban/what-is-kanban/
25. Doshi, P. (2018). *Agile Coach Toolkit #8: Limiting Work-In-Progress*. Dostupno na: https://www.scrum.org/resources/blog/agile-coach-toolkit-8-limiting-work-progress (Pristupljeno: 19 Oktobar 2020)
26. Gane, C. (2001). Process Management: Integrating Project Management and Development. In Tinirello, P.C. (Ed.) *New Directions in Project Management.* p 67-82. Boca Raton, FL: Auerbach Publications.
27. Hawrysh, S. i.Ruprecht, J. (2000). *Light Methodologies*: It's Like Déjà Vu All Over Again. Cutter IT Journal. 13: 4 - 12.
28. Herenda, M. (2016). *Agilne metodologije vođenja IT projekata kao unaprijeđenje u odnosu na standardne metodologije*. (Magistarski rad) Ekonomski fakultet Univerziteta, Sarajevo.
29. Hidalgo, E. (2019). *Adapting the scrum framework for agile project management in science: case study of a distributed research initiative*. *Heliyon*, *5*(3), 1-32. Dostupno na: https://www.cell.com/heliyon/fulltext/S2405-8440(18)34063-5?_returnURL=https%3A%2F%2Flinkinghub.elsevier.com%2Fretrieve%2Fpii%2FS2405844018340635%3Fshowall%3Dtrue (Pristupljeno: 12 August 2020)
30. Highsmith, J. (2004). *Agile Project Management*: creating innovative products, Addison Wesley

31. Hillaire, O. (2018). *Best Practices for Implementing a Hybrid Project Management Methodology*. Applied Information Management Program, University of Oregon, Eugene. (Magistarski rad). Dostupno na: https://scholarsbank.uoregon.edu/xmlui/handle/1794/24350 (Pristupljeno: 12 August 2020)
32. Hoda, R., Noble, J. i Marshall, S. (2010). *Organizing self-organizing teams. In: ACM/IEEE International Conference on Software Engineering*, 32, 2010, Cape Town, South Africa. Proceedings… Cape Town: [S.I.], may 2-8, 2010, 1, 285-294.
33. Institut za upravljanje projektom (2008). *A Guide to the Project Management Body of Knowledge*. Fourth Edition (PMBOK Guide). Newtown Square, PE: Project Management Institute.
34. ISO 2382 (1990). *Information processing systems - Vocabulary - Part 04: Organization of data*. Dostupno na: https://www.iso.org/standard/63598.html (Pristupljeno: 19 August 2020)
35. ISO 8402 (1994). *Quality Management and Quality Assurance – Vocabulary* (2th ed.). Dostupno na: https://www.iso.org/standard/20115.html (Pristupljeno: 17 August 2020)
36. IT Karijera. (2019). *IT sektor u BiH*. Dostupno na: https://itkarijera.ba/html-page/68/it-sektor-u-bih (Pristupljeno: 19 Juli 2020)
37. Jeffries, R. (2011). *What is Extreme Programming?*. Preuzeto 14. Augusta 2020, sa https://www.digite.com/kanban/what-is-kanban/
38. Kerzner, H. (2001). *Strategic Planning for Project Management using Project Management Maturity Model.* New York, NY: John Wiley & Sons.
39. Kniberg, H., Skarin, M., Poppendieck, M., i Anderson, D. (2010). *Kanban and Scrum: Making the most of both*. (1th ed.). USA. InfoQ.
40. Kumar, S. (2020). *Software Engineering | Spiral Model*. GeeksforGeeks. Dostupno na: https://www.geeksforgeeks.org/software-engineering-spiral-model/ (Pristupljeno: 10 Juni 2020)
41. Nerur, S., Mahapatra, R. K., i Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM*, *48*(5), 72–78.
42. *Oxford English Dictionary*. (1989). Oed.com. Dostupno na: https://www.dictionary.oed.com / (Pristupljeno: 6 Februar 2020)
43. Palmer, S. R., i Felsing, J. M. (2002). *A practical guide to feature-driven development*. (1th ed.). Prentice Hall PTR.
44. Pawlicka, A. (2021). *Agile software development process- Everything you need to know*. Dostupno na: https://selleo.com/blog/agile-software-development-process-everything-you-need-to-know (Pristupljeno: 6 Februar 2021)
45. Pereira, P., Torreão, P. i Maçal, A. S. (2007). *Entendendo Scrum para Gerenciar Projetos de Forma Ágil*. Mundo PM.
46. Petersen, K., Wohlin, C., i Baca, D. (2009). The waterfall model in large-scale development. International Conference on Product-Focused Software Process Improvement--PROFES 2009, 386-400.
47. Prašo, M., Junuz, E., i Hamulić, I. (2016). *Upravljanje softverskim projektima* [Ebook] (1st ed.). Univerzitet „Džemal Bijedić“ u Mostaru. Preuzeto 19. Jula 2020, sa http://up.fit.ba/ .

48. Project Management (2020). *3 Best Tools for Waterfall Project Management*. PMprojectmanagement.com. Dostupno na: https://project-management.com/3-best-tools-for-waterfall-project-management/ (Pristupljeno: 6 Februar 2020)
49. Project Management Institute. (2017). *A guide to the PROJECT MANAGEMENT BODY OF KNOWLEDGE (PMBOK Guide)* (6th ed. p.47). Pennsylvania: Project Management Institute.
50. Rahimian, V., i Ramsin, R. (2008). Designing an agile methodology for mobile software development: A hybrid method engineering approach. *2008 Second International Conference on Research Challenges in Information Science*.
51. Robinson, H., i Sharp, H. (2005). Extreme programming and Agile processes in Software Engineering. *The Social Side Of Technical Practices*, *3556*, 100-108.
52. Rosenberg, D., Stephens, M. (2003) Extreme Programming Refactored: the Case against XP. Apress, NY.
53. Rubin, K. (2013). *Essential Scrum*. Upper Saddle River, New Jersey: Addison-Wesley.
54. Schwaber, K. (1995). *Scrum Development Process*. OOPSLA'95 Workshop on Business Object Design and Implementation, Springer-Verlag.
55. Schwaber, K. (2004). *Agile Project Management with Scrum*, Microsoft Press.
56. Schwaber, K. (2007). *The enterprise and Scrum*. Microsoft Press.
57. Schwaber, K. i Beedle, M. (2002) *Agile Software Development with Scrum*. Prentice-Hall, Upper Saddle River.
58. Schwaber, K. i Sutherland, J. (2013). *The Scrum guide. The Definitive Guide* to Scrum: The Rules of the Game. Preuzeto 9. Augusta 2020, sa http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf#zoom=100..
59. Schwaber, K. i Sutherland, J. (2017). *The Scrum Guide*. Scrumguides.org. Dostupno na: https://www.scrumguides.org/ (Pristupljeno: 12 August 2020)
60. Scrum Alliance (2016). *The State of Scrum Report 2017Edition*. Dostupno na: https://www.scrumalliance.org/learn-about-scrum/state-of-scrum/2017-state-ofscrum (Pristupljeno: 15 August 2020)
61. Shenhar, A. J. i Dvir, D. (2007). *Reinventing project management: The diamond approach to successful growth and innovation.* Boston, MA: Harvard Business Press.
62. Shiklo, B. (2017). *What is Agile software development from customer's perspective?.* Dostupno na: https://www.scnsoft.com/blog/what-is-agile-software-development-from-customers-perspective (Pristupljeno: 8 Februar 2022)
63. Singh, V. (2021). *Crystal method in Agile.* Dostupno na: https://www.toolsqa.com/agile/crystal-method/ (Pristupljeno: 6 Februar 2021)
64. SlideUpLift. (2021). *Agile Development Cycle.* Dostupno na: https://slideuplift.com/powerpoint-templates/agile-methodology-01/ (Pristupljeno: 17 Februar 2021)
65. Stapleton, J. (1997). *Dynamic Systems Development Method Dsdm; the method in practice*. (1th ed.). Addison-Wesley.
66. Špundak, M. (2014). Mixed Agile/Traditional Project Management Methodology – Reality or Illusion?. *Procedia - Social And Behavioral Sciences*, *119*, 939-948.

67. Team, L. (2020) *7 things you need to know about feature driven development, Lvivity*. Preuzeto 12. Jula 2021, sa https://lvivity.com/7-things-about-feature-driven-development (Accessed: December 6, 2022).
68. Ured za trgovinu Vlade (2009). *Managing Successful Projects with PRINCE 2*. Norwich, UK: The Stationary Office.
69. Wells, D. (2013). *Extreme programming: A gentle introduction.* Dostupno na: http://www.extremeprogramming.org/index.html (Pristupljeno: 6 Februar 2020)
70. West, D., i Grant, T. (2010). *Agile Development: Mainstream Adoption Has Changed Agility for Application Development & Program Management Professional.* Cambridge, MA: Forrester Research, Inc.
71. Wysocki, R. (2014). *Effective Project Management* (7th ed.). New York: Wiley-Blackwell.
72. Zafar, I., Abbas, M. i Nazir, A. (2018). *The impact of Agile Methodology (DSDM) on Software Project Management*, CSL Press, USA, 1-6.

PRILOZI

Prilog 1. Lista kompanija

1. Ankora
2. Ant Colony
3. Atlantbh
4. Authority Partners
5. BS Telecom Solutions
6. Bytesoft
7. Claire Automotive Support
8. Comtrade
9. Endava
10. Evolt
11. HTEC Group
12. IDK Studio
13. Infinity Mesh
14. Infobip
15. Infostudio
16. Klika
17. Lanaco
18. Marvelsoft
19. MAUS
20. Meshmind
21. Ministry of Programming
22. Mistral
23. NetBreakPoint
24. Olive
25. Oracle
26. Orca
27. Propeller
28. QSS
29. Reppublika
30. ReSys
31. Royanis
32. Softray Solutions
33. Symphony
34. System Verification
35. Tacta
36. Techwave
37. Virgin Pulse
38. WizardHealth
39. Waltercode
40. ZenDev
41. ZIRA

Prilog 2. Anketna pitanja

1. Koliko godina postoji Vaša kompanija?
    a) 1-10
    b) 11-20
    c) 21-30
    d) 31-50
    e) 51+

2. Koliko zaposlenika broji Vaša kompanija?
    a) Manje od 10 zaposlenih
    b) 10-49 zaposlenih
    c) 50-249 zaposlenih
    d) 250+

3. Kojoj starosnoj dobi pripadate?
    a) 18-25
    b) 26-35
    c) 36-45
    d) 46-55
    e) 55+

4. Koliko godina iskustva imate u IT industriji?
    a) <1 godine
    b) 1-5 godina
    c) 5-10 godina
    d) 10+ godina

Oblast metodologije

5. Koju od navedenih metoda projektnog menadžmenta većinski dio Vaše kompanije primjenjuje na poslovnim projektima?
    a) Tradicionalnu
    b) Agilnu

Oblast agilne metodologije

6. Koliko već dugo vaša kompanija prakticira agilni način rada?
    a) <1 godine
    b) 1-2 godinu
    c) 2-5 godina
    d) 5+ godina

7. Da li se slažete da IT projekti ukazuju na veću potrebu primjene agilne metodologije u odnosu na tradicionalnu zbog: fleksibilnosti mijenjanja zahtjeva tokom razvoja projekta; brže isporuke softvera; veće kolaboracije sa klijentom I veće transparentnosti među članovima tim?
    a) DA
    b) NE

8. Ukoliko je Vaš odgovor bio "NE" na prethodno pitanje, koje biste Vi faktore naveli zbog kojih IT projekti ukazuju na veću potrebu primjene agilne umjesto tradicionalne metodologije?

    _____

9. Koju od navedenih 4 stavke biste istakli kao najznačajniju tokom razvoja projekta?
    a) Fleksibilnost mijenjanja zahtjeva tokom razvoja projekta
    b) Brža isporuke softvera
    c) Veća kolaboracija sa klijentom
    d) Veća transparentnost među članovima tima

10. Koje biste od navedenih stavki izdvojili kao najistaknutije prednosti agilne metodologije u odnosu na tradicionalnu?
    Možete odabrati više ponuđenih odgovora.
    a) Bolja prilagodba promjenama na tržištu
    b) Smanjenje rizika i bolja kontrola nad projektom
    c) Smanjenje broja grešaka i problema u projektima
    d) Veća korisnička vrijednost zbog isporuke manjih iteracija
    e) Povećana motivacija tima
    f) Efikasnije resursno upravljanje
    g) Niži stepen neizvijesnosti finalnog proizvoda
    h) Kraće vrijeme za planiranje

i) Stvara pogodno okruženje za inovacije
j) Drugo:_____

11. Da li su se desila određena poboljšanja u navedenim stavkama uvođenjem agilne metode u Vašu kompaniju?

	Mnogo poboljšano	Poboljšano	Nema promjene	Gore	Mnogo gore	Ne znam
Brža isporuka i kvalitet softvera						
Kolaboracija sa klijentom						
Fleksibilnost na izmjene zahtjeva klijenta						
Produktivnost i transparentnost među članovima tima						
Smanjenje rizika projekta						
Ubrzan odgovor na tržišne trendove						
Vrijeme za planiranje i prikupljanje zahtijeva						

12. Koji od navedenih faktora predstavljaju prepreke pri uvođenju agilne metodologije? Možete odabrati više ponuđenih odgovora.
    a) Izazov u izmjeni organizacijske kulture
    b) Izazov uklapanja agilnih elemenata u prethodni ustaljeni tradicionalni okvir
    c) Sveopći otpor promjenama
    d) Nedovoljna podrška menadžmenta
    e) Nedostatak individualaca sa adekvatnim znanjem i vještinama iz oblasti agilnih metodologija
    f) Kompleksnost projekta
    g) Proiciranje dodatnih troškova obuke uposlenih
    h) Nije bilo prepreka
    i) Drugo:_____

13. Planirate li za buduće projekte primjenjivati agilni način rada ?
    a) Da
    b) Nisam siguran/na

c) Ne

14. Koja agilna metodologija je najzastupljenija u Vašoj kompaniji?
    a) Scrum,
    b) Ekstremno programiranje (XP)
    c) „Feature driven“ razvojni okvir (FDD)
    d) Kanban,
    e) Dynamic system razvojni okvir (DSDM)
    f) „Crystal“ razvojni okvir

15. Koji su razlozi zbog kojih primjenjujete Scrum metodologiju?
    Možete odabrati više ponuđenih odgovora.
    a) Oslanja se na jasno definisane uloge (Scrum Master, Product Owner, i Development Team)
    b) Bolje planiramo i prioritiziramo zadatke kroz Product Backlog i Sprint Backlog
    c) Dobro se uklapa sa našom timskom dinamikom
    d) Bolje upravljamo promjenama i nepredvidivim situacijama
    e) Zbog pozitivnog iskustva sa Scrumom u prethodnim projektima
    f) Pomaže nam da redovno procenjujemo i poboljšavamo naše procese kroz retrospektive
    g) Ima dobru ravnotežu između fleksibilnosti i strukture
    h) Ostalo: _____

Oblast tradicionalne metodologije

16. Koju tradicionalnu metodologiju primjenjujete u Vašoj kompaniji?
    a) Vodopadni model
    b) Spiralni model

17. Razlozi zbog kojih još uvijek koristite tradicionalnu metodologiju?
    Možete odabrati više ponuđenih odgovora.

    a) Nedostatak razumijevanja ili uvjerenja u prednosti agilne metodologije
    b) Nedostatak resursa (vremena, financija, alata) za prelazak na agilnu metodologiju
    c) Nedostatak iskustva ili obuke za primjenu agilne metodologije
    d) Struktura organizacije koja nije kompatibilna s agilnom metodologijom
    e) Otpor promjenama među članovima tima ili organizacije
    f) Učestali projekti koji zahtijevaju tradicionalnu metodologiju
    g) Nedostatak podrške višeg menadžmenta za prelazak na agilnu metodologiju
    h) Tradicionalna metodologija se smatra pouzdanijom ili efikasnijom
    i) Zadovoljni smo postojećom metodologijom
    j) Drugo: _____

18. Šta smatrate najvećim nedostacima tradicionalne metodologije?
    Možete odabrati više ponuđenih odgovora.
    a) Spora i nefleksibilna prilagodba promjenama u zahtjevima
    b) Dugotrajni ciklusi razvoja i isporuke
    c) Visoka kompleksnost i formalni procesi

d) Povećani rizici od grešaka i neuspjeha projekta
e) Slaba sposobnost brze reakcije na tržišne promjene
f) Nedostatak interakcije s klijentima tijekom razvoja
g) Prekomjerna dokumentacija i administracija
h) Nemogućnost na samom početku procijeniti sve potrebe klijenta
i) Drugo:_____

19. Smatrate li da biste unaprijedili Vašu kompaniju prelaskom na agilni način rada?
Možete odabrati više ponuđenih odgovora.
a) Da
b) Ne

20. Da li imate u planu preći na agilni način rada?
Možete odabrati više ponuđenih odgovora.
a) Da
b) Ne

21. Ukoliko je Vaš odgovor na prethodno pitanje bio "Da", koju biste agilnu metodologiju uveli?
a) Scrum,
b) Ekstremno programiranje (XP)
c) „Feature driven“ razvojni okvir (FDD)
d) Kanban,
e) „Dynamic system“ razvojni okvir (DSDM)
f) „Crystal“ razvojni okvir