Usporedba tradicionalnog i agilnog načina vođenja projekata u razvoju softverskih proizvoda

Marić, Luka

Sveučilište u Zagrebu

Ekonomski fakultet

Diplomski sveučilišni studij Analiza i poslovno planiranje

USPOREDBA TRADICIONALNOG I AGILNOG NAČINA VOĐENJA PROJEKATA U RAZVOJU SOFTVERSKIH PROIZVODA

Diplomski rad

Luka Marić

Zagreb, ožujak 2020.

Sveučilište u Zagrebu

Ekonomski fakultet

Diplomski sveučilišni studij Analiza i poslovno planiranje

USPOREDBA TRADICIONALNOG I AGILNOG NAČINA VOĐENJA PROJEKATA U RAZVOJU SOFTVERSKIH PROIZVODA

COMPARISON OF TRADITIONAL AND AGILE WAY OF PROJECT MANAGEMENT IN SOFTWARE DEVELOPMENT

Diplomski rad

univ. bacc. oec. Luka Marić, 0067523427

Mentor: Prof. dr. sc. Mislav Ante Omazić

Zagreb, ožujak 2020.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Izjavljujem i svojim potpisom potvrđujem da je završni/diplomski/specijalistički rad, odnosno doktorska disertacija isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju korištene bilješke i bibliografija. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Student/ica:

U Zagrebu, 10.03.2020.

Luka Kovač
(potpis)

SAŽETAK I KLJUČNE RIJEČI NA HRVATSKOM JEZIKU

Svako poduzeće koje se bavi razvojem nekog oblika softverskih proizvoda mora usvojiti određenu, dobro definiranu, metodologiju tj. upravljački okvir za razvoj takvih proizvoda. Te projektne metodologije olakšavaju upravljanje procesima u proizvodnji softvera visoke kvalitete. Procesi razvoja softverskih proizvoda se uglavnom sastoje od sljedećih faza: detaljna analiza zahtjeva, dizajn, implementacija, testiranje, puštanje u produkciju i održavanje. U svijetu razvoja softverskih proizvoda, danas postoje dva najpopularnija načina upravljanja projektima: tradicionalni i agilni način razvoja. Tradicionalna metodologija i modeli proizašli iz nje zahtijevaju dobro osmišljen plan, slijedni napredak faza te definiran skup zahtjeva za razliku od agilnih načina koji imaju manje stroge smjernice, a karakterizira ih iterativan i inkrementalan pristup razvoju proizvoda. Agilni načini tako dozvoljavaju neodređeni broj repeticija pojedinih faza koje jednom kada su završene rezultiraju upotrebljivim inkrementom proizvoda s novim setom funkcionalnosti. Cilj ovoga rada je opisati vodopadni model kao najpopularniji oblik primjene principa tradicionalne metodologije te Scrum upravljački okvir kao najpopularniji oblik primjene principa agilnog načina razvoja proizvoda. Opisom projekata iz poslovne prakse pokušat će se pokazati kako se teorijski principi primjenjuju u praksi te postoje li neki faktori na koje bi se trebala obratiti posebna pažnja prilikom odlučivanja o primjeni pojedine metodologije vođenja projekta.

Ključne riječi: projektni menadžment, softverskih inženjering, metodologije razvoja softverskih proizvoda, tradicionalna metodologija, agilne metodologije, vodopadni model, Scrum

SAŽETAK I KLJUČNE RIJEČI NA ENGLESKOM JEZIKU

Each company engaged in the development of some form of software products must adopt a specific, well-defined methodology, i.e.management framework for the development of such products. These project methodologies facilitate the management of processes in the production of high quality software. Software product development processes mainly consist of the following stages: detailed requirements analysis, design, implementation, testing, production release and maintenance. In the world of software product development, there are two most popular ways of managing projects today: the traditional and the agile methods of development. The traditional methodology and models derived from it require a well-designed plan, sequential phase progress, and a defined set of requirements, as opposed to agile modes that have less stringent guidelines, characterized by an iterative and incremental approach to product development. The agile development thus allows for an indefinite number of repetitions of individual stages that, once completed, result in a usable product increment with a new set of functionalities. The aim of this paper is to describe the waterfall model as the most popular form of applying the principles of the traditional methodology and the Scrum framework as the most popular form of applying the principles of the agile product development. The description of the projects in business practice aims to show how the theoretical principles are implemented in practice and whether there are some factors that should be paid special attention to when deciding on the application of a particular project management methodology.

Key words: project management, software engineering, software development methodologies, traditional software development methodology, agile methodologies, waterfall model, Scrum

Sadržaj

1. UVOD

Softverski proizvodi su u modernom poslovanju postali ključna komponenta koja često može odrediti smjer i uspješnost poslovanja čitavog poduzeća. Stoga izrada softverskih proizvoda postaje jedan od nezanemarivih elemenata prilikom kretanja u nove poduzetničke pothvate, prilagodbe postojećeg poslovnog rješenja prilikama na tržištu ili prilikom ekspanzije na nova tržišta i tržišne segmente. Poduzeća imaju dvije mogućnosti, razvijati i održavati softver unutar vlastitih radnih kapaciteta ili unajmiti dedicirana poduzeća koja kroje softverska rješenja po narudžbi. Od samih početaka razvoja softvera do danas su se procesi i kompleksnost razvoja softvera značajno promijenili. Stoga su s vremenom nastale i specifične metodologije razvoja kako bi se standardizirali procesi i povećala produktivnost projektnih timova kojima je zadatak proizvodnja takvog tipa proizvoda.

1.1. Predmet i cilj rada

Metodologije u projektnom menadžmentu se obično definiraju kao skupine metoda, tehnika, procedura, pravila, predložaka i najboljih praksi koje se koriste na projektu (Šundak, 2014). One služe osobama zaduženim za uspostavu i izvršenje projekata kao vodilje u kreiranju i praćenju projektnih aktivnosti kako bi se maksimizirao njihov ishod te zadovoljile sve interesne strane uključene u projekt. Predmet ovoga rada biti će projekti razvoja softverskih proizvoda. U informacijsko-komunikacijskoj industriji, projekte razvoja proizvoda možemo podijeliti na one vođene po principima tradicionalne metodologije te modela proizašlih iz nje i agilnog načina razvoja i metodologija te modela i upravljačkih okvira proizašlih iz smjernica koje on nalaže. Tradicionalna metodologija zahtijeva dobro osmišljen plan te dobro definiran skup zahtjeva, za razliku od agilnih načina koje karakterizira iterativni pristup razvoju, manje formalne smjernice te lakša prilagodivost projektnih procesa.

Cilj teorijskog dijela rada je, korištenjem znanstvene literature, definirati ulogu i značaj projektnog menadžmenta u razvoju softverskih proizvoda te nastavno na njega prikazati ključne karakteristike primjene tradicionalnog i agilnog načina u vođenju projekata razvoja softverskih proizvoda. Kroz empirijski dio rada pokušat će se opisati ključne karakteristike dvaju najpopularnijih modela razvoja na primjerima projekata iz prakse te identificirati pitanja na koja bi se trebalo odogovoriti prije odabira načina vođenja projekta.

1.2. Izvori podataka i metode prikupljanja

U ovom radu se koriste primarni i sekundarni izvori podataka. Sekundarni izvori podataka se sastoje od raznolike znanstvene literature s temom projektnog menadžmenta te upravljanja procesima u razvoju softverskih proizvoda. Osim znanstvene literature, koristi se i nekolicina članaka i izvora s web stranica koji sadrže informacije povezane s temom rada kako bi se prikazali najažurniji komentari i statistike vezane uz temu.

Za potrebe pisanja rada, koristit će se dva primarna izvora podataka. Prvi izvor je provedeni dubinski intervju s projektnim menadžerom zaduženim za vođenje projekata u jednoj od najuspješnijih digitalnih agencija u Hrvatskoj koja se bavi razvojem softverskih proizvoda. Kroz dubinski intervju će se ispitati kako su opseg, vrijeme, budžet i drugi faktori utjecali na odabir metodologije i strukturu procesa u vođenju dva projekata razvoja softtverskih proizvoda. Drugi izvor podataka bit će jednostavni anketni upitnik koji je postavljen članovima projektnih timova spomenutih dvaju projekata. Usporedbom rezultata anketnih upitnika pokušat će se donijeti zaključak o uspješnosti primjene pojedine projektne metodologije na opće zadovoljstvo projektnih članova.

1.3. Sadržaj i struktura rada

U prvom dijelu rada opisat će se projektni menadžment kao područje unutar organizacijske teorije i prakse. S obzirom na temu rada, dodatno će se opisati važnost projektnog menadžmenta u procesima razvoja softverskih proizvoda te kako su pojedine karakteristike informacijsko-komunikacijske industrije i globalni trendovi utjecali na razvitak projektnog menadžmenta. Centralni dio rada obuhvatit će dva najpopularnija načina vođenja projekata u modernom razvoju softverskih proizvoda, vodopadni model te Scrum upravljački okvir. U tom dijelu rada će se nastojati pregledom relevantne znanstvene literature prikazati teorijski okvir i kontekst tradicionalne metodologije te njenih najpopularnijih modela i agilnog načina razvoja softverskih proizvoda te metodologija i upravljačkih okvira proizašlih iz smjernica agilnog manifesta.

U završnom dijelu rada će se opisnom studijom slučaja opisati projektni procesi i karakteristike na primjerima dvaju projekata. Jedan je vođen po principima tradicionalne metodologije tj. vodopadnim modelom, dok je drugi projekt vođen po principima agilnog načina razvoja tj. kojemu je Scrum upravljački okvir pomogao u definiranju uloga članova projektnog tima te projektnih procesa.

Zaključno, s obzirom na informacije dobivene iz rezultata intervjua s projektnim menadžerom te na informacije dobivene iz anketnih upitnika, pokušat će se identificirati ključna pitanja na koja bi se trebalo odgovoriti prije odabira projektne metodologije te ima li odabir pojedinog načina vođenja projekta utjecaj na opće zadovoljstvo projektnih članova uspjehom projekta i radnim uvjetima na njemu.

2. PROJEKTNI MENADŽMENT U RAZVOJU SOFTVERSKIH PROIZVODA

U ovom poglavlju rada, definirat će se projektni menadžment počevši s definicijom samog projekta te nastavno na nju definiranjem uloga projektnog menadžmenta u razvoju softverskih proizvoda. S obzirom na to da da se karakteristike i zadaci projektnih menadžera razlikuju s obzirom na karakteristike pojedinih industrija i zahtjeva za funkcionalnostima proizvoda koje se unutar njih grade, u ovom dijelu rada će se obraditi karakteristike softverskog inženjerstva i način na koji su se projekti i projektni timovi prilagodili njegovom kontekstu.

2.1. Pojmovno određenje projektnog menadžmenta

Projektni menadžment predstavlja područje unutar organizacijske teorije i prakse koje se konstantno razvija. Koncept projektnog menadžmenta predstavlja sustavni pristup efikasnom menadžmentu preko optimalizacije veza, informacija, odluka, dokumentacije i aktivnosti u svim fazama životnog ciklusa projekta (Vlahov, 2013). Kako bi se uspješno objasnio pojam projektnog menadžmenta, potrebno je započeti s definicijom projekta.

2.1.1. Karakteristike projekta

Kerzner (2017) projektom smatra svaki povezani niz aktivnosti koji:

- Ima specifičan cilj koji se mora ostvariti prateći određene specifikacije;
- Ima definiran početak i kraj;
- Ima financijska ograničenja;
- Za svoju provedbu troši ljudske i ne ljudske resurse (npr. novac, rad, oprema).

Prije nego li se krenu opisivati karakteristike, potrebno je opisati prvi dio Kerznerove definicije, a to je projekt kao povezani niz aktivnosti.

2.1.1.1. Povezani niz aktivnosti u projektu

Povezanost implicira na to da među aktivnostima na projektu mora postojati logička i tehnička veza. Postoji redoslijed kojim se aktivnosti na projektu moraju odvijati kako bi projekt bio završen. Aktivnosti se smatraju povezanima jer je proizvodni učinak jedne aktivnosti ujedno i ulaz[1] koji je potreban da bi sljedeća aktivnost započela. Na primjer, potrebno je dizajnirati računalni program prije nego što ga se može početi programirati. S

[1] eng. *input*

druge strane, možemo imati i nepovezane aktivnosti koje moraju biti završene kako bi neki projekt bio gotov. Uzmimo za primjer bojanje interijera kuće. Uz neke iznimke, sobe mogu biti obojene u bilo kojem redoslijedu, ali interijer kuće neće biti završen dokle god sve sobe u kući nisu obojene. Stoga, bojanje interijera kuće podrazumijeva slijed aktivnosti, ali se ono ne može smatrati projektom ako uzmemo u obzir prethodno navedenu definiciju projekta (Wysocki, 2011).

2.1.1.2. Specifičan cilj projekta

Svaki projekt mora imati specifičan cilj. Tako na primjer, mnoga poduzeća unajmljuju agencije za izradu web stranica, ali je svaka web stranica posebna i drugačija jer je nastala kao rezultat posebnih zahtjeva.

Međutim, Wysocki (2011) iznosi kako projekti većega obuhvata mogu biti podijeljeni u nekoliko potprojekata od kojih se svaki može smatrati samostalnim projektom iako imaju zajednički specifičan stil. Podjela se vrši zbog lakše i bolje menadžmentske kontrole. Na primjer, potprojekti mogu biti definirani na razini funkcionalne jedinice, odjela ili geografske lokacije. Prema Wysockiju, ova dekompozicija složenih projekata u potprojekte se radi s ciljem pojednostavljenja upravljanja rasporedom resursa te smanjuje potrebu za komunikacijom među odjelima dok se radi na određenoj aktivnosti. Nedostatak je taj što su projekti u tom slučaju samostalni. Iako samostalnost projektima dodaje još jedan nivo kompleksnosti, svakako se njome može upravljati.

2.1.1.3. Vremenski rok projekta

Projekti imaju određeni rok izvršenja. Taj datum može biti nametnut od strane menadžmenta ili eksterno specificiran od strane klijenta, vladine agencije ili nekog drugog tijela za kojeg se projekt provodi. Ono što je bitno jest da nitko od razvojnog tima nema kontrolu nad tim rokom. Projekt bi trebao završiti na specificiran datum bez obzira na to da li je sve iz obuhvata projekta dovršeno ili ne (Wysocki, 2011).

U praksi će se često vremenski rok produživati dokle god projekt ne bude udovoljavao prethodno definiranim zahtjevima jer bi u protivnom rezultat projekta bio nedovršeni ili poludovršeni proizvod koji bi mogao prouzrokovati frustracije prilikom korištenja te naštetiti reputaciji proizvođača tj. poduzeća koje je provodilo projekt.

2.1.1.4. Financijska ograničenja i resursi projekta

Resursi podrazumijevaju ljude, strojeve, softver i druga sredstva neophodna za završavanje projekta. Ovi resursi se mogu prilagođavati ovisno o menadžmentu, no smatraju se fiksnim resursima jednom kada je projekt dogovoren i kada njim krene upravljati projektni menadžer. Uzmimo za primjer da poduzeće u danom trenutku ima samo jednog raspoloživog grafičkog dizajnera. On predstavlja fiksni resurs koji je dostupan projektnim menadžerima. Ako je projektu dodijeljen samo jedan dizajner, to je onda resurs na kojeg projektni menadžer ne može utjecati već samo viši menadžment (Wysocki, 2011).
Financijska ograničenja odnosno budžet, dodatno su opisani u poglavlju „Ograničenja projekta".

2.1.1.5. Projektna specifikacija

Prethodno navedene odrednice projekata poklapaju se i prema Kerzneru (2017) i prema Wysockiju (2011), međutim, Wysocki definira još jednu karakteristiku projekata, a ta je da su vođeni prema određenoj specifikaciji. Prema njemu, klijent očekuje izvršenje projekta uz određene funkcionalnosti i kvalitetu. Ti zahtjevi mogu biti samo-nametnuti kao što je npr. datum izvršetka projekta, a mogu biti i posebno specificirani od strane klijenta kao što bi, primjerice, bila izrada izvještaja o prodaji na tjednoj bazi. Iako bi projektni menadžeri trebali projektnu specifikaciju tretirati fiksnom, u stvarnosti brojni faktori za vrijeme trajanja projekta utječu na njene promjene. To se događa u prilikama kada neki zahtjevi nisu dovoljno detaljno definirani od strane klijenta ili se, primjerice, promijenila poslovna situacija (što je čest slučaj kod projekata s dužim rokom). Nerealno je očekivati da se specifikacija neće mijenjati kroz životni ciklus projekta. Sistemske specifikacije se mogu i hoće mijenjati te zbog toga predstavljaju poseban izazov projektnim menadžerima u dostizanju projektnih ciljeva.

2.1.1.6. Nesigurnosti na projektu

Budući da je svaki projekt jedinstven ponekad je teško jasno definirati ciljeve projekta, procijeniti točno koliko će vremena trebati da se projekt završi i odrediti koliko će to sve koštati. Vanjski čimbenici također uzrokuju neizvjesnost, poput dobavljača koji prestaje s poslovanjem ili člana tima koji treba uzeti neplanirani godišnji odmor/bolovanje. Nesigurnost

je tako jedna od ključnih karakteristika projekata zbog koje je upravljanje projektima toliko izazovno (Schwalbe, 2015).

2.1.2. Ograničenja projekta

Ograničenja projekta se postavljaju od strane projektnih sponzora/ključnih dionika ili odbora projekata tako da postoje zajednička, dogovorena očekivanja od projekta. Usko su vezana s navedenim karakteristikama projektima jer se zapravo svaka karakteristika predstavlja ograničenje na koje bi sponzori projekta trebali obratiti pozornost prije kretanja s projektom.
Prema Wysockiju (2011) i Siegelaub (2007), ovo su šest temeljnih ograničenja projekta:

- Obuhvat,
- Kvaliteta,
- Budžet,
- Vrijeme,
- Resursi,
- Rizik.

Ova ograničenja formiraju nezavisan set. Promjena u jednom ograničenju može dovesti do potrebe mijenjanja drugog kako bi se projekt vratio u ravnotežno stanje. U tom kontekstu, navedeni parametri formiraju sustav koji mora ostati nepromijenjen kako bi projekt bio balansiran zbog tolike važnosti svakog pojedinog ograničenja na krajnji uspjeh/neuspjeh projekta (Wysocki, 2011).
Iako se ograničenja preklapaju prema Wysockiju i Siegelaub, Siegeleaub (2007), a i brojna druga literatura, dodatno navodi i rizik. Prema njemu, potrebno je prethodno definirati rizik koji je prihvatljiv projektnim dionicima tj. sponzorima. Ako projektni menadžer ne uspije kontrolirati veće rizike (mitigacija/transfer rizika), projektni dionici tj. sponzori moraju odlučiti je li im prihvatljivo nastaviti s većom izloženošću riziku ili će se projekt zatvoriti.

2.1.2.1. Obuhvat kao ograničenje projekta

Obuhvat je iskaz kojim su definirane granice projekta. On ne ukazuje samo na to što je obuhvaćeno projektom, već i na ono što u sklopu projekta neće biti napravljeno. U industriji informacijskih tehnologija, obuhvat je često definiran dokumentom zvanim *funkcijska specifikacija.* Ovisno o industriji, može poprimiti različita imena, no bez obzira kako bio nazvan, ovaj dokument predstavlja temelj za sav rad koji slijedi. Nije tajna da se obuhvat projekta može promijeniti, ali je važno tu promjenu detektirati na vrijeme i donijeti odluku o

tome kako prilagoditi projekt promjeni. Upravo to je jedan od najzahtjevnijih zadataka s kojima se susreću projektni menadžeri prilikom vođenja projekata (Wysocki, 2011).

Obuhvat se može prezentirati kao raspon. Siegelaub (2007) to objašnjava na sljedeći način kada komunicira projektnom menadžeru:
„*Želim da se napravi proizvod A, on je obavezan. Ako ostane viška vremena i novca, također bi htio proizvod B. Ako se proizvod B ne može napraviti, imati proizvod A će biti zadovoljavajuće*."
Nadalje, Siegelaub (2007) navodi da s ovakvom tolerancijom na obuhvat, projektni menadžer ima fleksibilnost oko onoga što treba biti isporučeno, ali u okviru striktnih i dogovorenih limita. Takva tolerancija može biti prezentirana i kao negativan raspon, primjerice:
„Želim da se razviju proizvodi C i D, ali ako pred kraj budete kratki s vremenom i novcem, prihvatljivo je da se proizvod D ne razvije."
Ovakve situacije se javljaju kada su definirane osnovne stavke koje se projektom moraju dostaviti. Ostale, sporedne stavke se mogu dostaviti kasnije bez ugrožavanja ključnih projektnih ciljeva i bez disrupcija projektnog obuhvata.

2.1.2.2. Kvaliteta kao ograničenje projekta

Ograničenje kvalitete je dosta slično ograničenju obuhvata, samo što je fokus kvalitete na karakteristikama isporučenih proizvoda. Tolerancija na kvalitetu djeluje po principima točnosti. Ona predstavlja stupanj do kojega razvijena stavka odgovara definiranim karakteristikama. Primjerice, ako je projektni zahtjev da se napravi stol dug točno jedan metar, tada bi isporučeni stolovi koji su dugi 99 cm ili 101 cm bili odbijeni. Alternativno, ako je postavljenja tolerancija od +/- 1 cm, to znači da bi se mogli isporučiti proizvodi dimenzija 99,5 ili 100,5 cm te bi i dalje bili prihvatljivi jer ispunjavaju postavljene kriterije kvalitete (Siegelaub, 2007).

Ako promotrimo ostala projektna ograničenja, u mnogim klasičnim situacijama, kada su vremenski rokovi, budžeti i resursi iscrpljeni, kvaliteta proizvoda je ta koja najčešće pati kako bi se projekti završili unutar definiranih okvira. Voditelji projekata tako mogu i isporučiti definirane stavke, ali ih se možda nije stiglo temeljito istestirati ili se zbog spomenutih okolnosti moralo odustati od pojedinih definiranih karakteristika proizvoda što svakako predstavlja prijetnju i potencijalnu opasnost prilikom vrednovanja kvalitete proizvoda.

Wysocki (2011) definira dva tipa kvalitete koji su dio svakog projekta:

- Kvaliteta proizvoda – Kvaliteta krajnjeg produkta nastalog tijekom projekta. Za kontrolu proizvoda koriste se alati kao što su: planiranje kvalitete, održavanje kvalitete i kontrola kvalitete.
- Kvaliteta procesa – Odnosi se na kvalitetu procesa projektnog menadžmenta. Fokus je na tome koliko dobro procesi djeluju i kako ih se može poboljšati. Alati koji se koriste su neprekidno poboljšavanje kvalitete i upravljanje kvalitetom procesa.

Nadalje, Wysocki iznosi kako je dobar program upravljanja kvalitetom, s postavljanim procesima koji imaju zadaću monitorirati sav posao na projektu, uvijek dobra investicija. Ne samo da doprinosi zadovoljstvu klijenta, već i pomaže poduzećima da učinkovitije koriste svoje resurse. Upravljanje kvalitetom je tako jedno od područja koje svakako ne bi trebalo biti ugroženo.

2.1.2.3. Budžet kao ograničenje projekta

Budžet je jedna od najvažnijih stavki o kojima projektni menadžer mora voditi računa kroz čitav životni ciklus projekta. Uzima se u obzir već kod rane i neformalne faze projekta. Ovisno o tome koliko je naručitelj projekta proveo vremena definirajući i razmišljajući o projektu, toliko inicijalna brojka može biti blizu odnosno daleko od stvarnog troška projekta. Projektni menadžeri se često nalaze u situaciji u kojoj je naručitelj spreman potrošiti točno određeni iznos sredstava te je u takvim slučajevima na projektnim menadžerima zadatak da jednostavno rade s onim što imaju. S druge strane, u formalnijim situacijama, na projektnim menadžerima je zadatak pripremiti ponudu za projicirani posao. U toj ponudi bi se trebala uključiti procjena sveukupnih troškova projekta. Takva ponuda omogućava naručitelju projekta/klijentu lakše donošenje odluke o tome hoće li projekt zaživjeti ili ne (Wysocki 2011).

Na budžet i upravljanje budžetom značajno utječu i odabrane projektne metodologije koje daju okvir za vođenje projekta. Budžetna ograničenja i prakse korištenja budžeta koji su karakteristični za jednu metodologiju, ne moraju biti isti i za drugu. O tome će se raspravljati u poglavljima koja se odnose na projektne metodologije.

2.1.2.4. Vrijeme kao ograničenje projekta

Wysocki (2011) iznosi kako naručitelj projekta treba specificirati vremenski okvir i krajnji rok do kada projekt treba biti završen. U određenoj mjeri, vrijeme i trošak su u inverznom odnosu. Jednom kada je obuhvat projekta definiran, vrijeme mu se može reducirati, ali se u tom slučaju troškovi povećavaju. Prema Wysockiju, vrijeme je zanimljiv resurs. S obzirom na to da ga ne možemo skladištiti, troši se nevezano za to ~~k~~konzumiramo li ga ili ne. Na projektnim menadžerima je zadatak da obraćaju pažnju na nadolazeće vrijeme te ga prigodno alociraju na što efikasniji i efektivniji način. Buduće vrijeme je resurs koji se može razmjenjivati unutar projekta pa čak i među različitim projektima. Jednom kada projekt započne, vrijeme je primarni resurs kojeg projektni menadžeri koriste kako bi održavali projekt unutar definiranog rasporeda.

Prema Siegelaub (2007) vrijeme i budžet su „najopipljivije“ mjere tj. ograničenja projekta. Upravo zato su i prvo mjesto gdje sponzori gledaju kada se projekt nađe u problemima, a često znaju biti i jedino mjesto na koje se u tom slučaju obraća pozornost. Često su sva ostala ograničenja u potpunosti zanemarena od strane projektnih sponzora dokle god se projekt završava u okviru budžeta i na vrijeme. Zato je čest slučaj da se prikrivaju neke nedostajuće značajke ili se previđa kontrola kvalitete. Budući da čak ni sponzori obično nisu sigurni koje su definirane karakteristike u obuhvatu projekta ili što se radi kontrolom kvalitete, svi budu jednako sretni ako se one ignoriraju ili smanje na najmanju moguću mjeru, u korist vremena i troškova.

2.1.2.5. Resursi kao ograničenje projekta

Prema Wysockiju (2017) resursi mogu biti ljudi, oprema, fizička postrojenja ili skladišta koja imaju ograničavaju raspoloživost. Mogu biti dodijeljeni iz resursa poduzeća ili unajmljeni od treće strane. Hartney (2017) svrštava svaki projektni resurs u jednu od sljedećih kategorija, s obzirom na to kako im se dodjeljuju projektni troškovi:

- Direktni - resursi čije korištenje rezultira gotovim proizvodnim jedinicama. Na primjer, sat rada radnika ili broj utrošenih komada određenog materijala. Stvarni trošak bi se trebao podijeliti na razuman broj projekata kojima će se direktni resurs dijeliti. Uz direktne resurse bi se trebali vezati i sekundarni troškovi kao što su radne beneficije, davanja za mirovine, bonusi ili bilo koji drugi trošak povezan s resursom.

- Indirektni - odnosi se na posao koji je potreban kako bi se proizveo krajnji proizvod, ali nije direktno uključen u proizvodnju. Ovdje se podrazumijevaju stvari kao što su: kontrola kvalitete, nadzor proizvodnje i projektni menadžment. Ovu vrstu resursa svakako treba uključiti u procjene prilikom računanja projektnih troškova. Najbolja praksa je da se prikladno alociraju odgovarajućim zadacima.
- Neizravni - administrativni troškovi poput plaće izvršnog direktora ne mogu se pripisati projektu, ali ovisno o organizacijskog strukturi, ih ponekad moraju platiti svi projekti. Razlika između resursa koji rezultiraju indirektnim i neizravnim troškovima je u tome da se indirektni troškovi mogu direktno pripisati projektu (pr. projektni menadžment), dok neizravni (pr. rad izvršnog direktora) ne moraju nužno imati ikakvu uključenost u projekt.

U svakom slučaju, resursi su centralni alat planiranja projektnih aktivnosti bez kojih je nemoguće dovršiti projekt. U industriji razvoja softverskih proizvoda najbitniji resurs su ljudi, zatim slijede računalna infrastruktura i oprema koju se koristi u razvoju softvera.

2.1.2.6. Rizik kao ograničenje projekta

Kako bi se upravljalo rizikom kao ograničenjem, poduzeća moraju pronaći razinu tolerancije na rizik koja je prihvatljiva svim dionicima. Primjerice, ako pojedini dobavljač podbaci i ne može više nastaviti pružati svoje usluge, potrebno je pronaći novoga dobavljača u rasponu od cijene X, vremena potrebnog za dostavu proizvoda/usluga Y i kvalitete Z. Uspostavljanjem razine tolerancije, dionicima na projektu je lakše odrediti koliki su rizik spremni preuzeti u situacijama sličnim navedenom primjeru. Drugi način sagledavanja rizika može biti kroz neočekivane prilike koje se mogu pojaviti. Iskorištavanje novonastale prilike će, naravno, uključivati i rizik, stoga je korisno pokazati scenarije svojim dionicima i odrediti njihov prag tolerancije. Ako se, recimo, pojavi prilika da se osvoji veći tržišni udio, potrebno je odgovoriti na pitanja hoće li dionici biti spremni povećati iznos ulaganja i koje bi bile granice njihovog ulaganja (Shenoy, 2017).

Upravo zadnji primjer (prilika osvajanja većeg tržišnog udjela) ukazuje na to da se rizik ne mora vezati isključivo uz negativne scenarije i prijetnje poduzeću/projektu, već da treba uključiti i prilike koje mogu rezultirati pozitivnim rezultatom.

2.1.3. Projektni menadžment

Projektni menadžment uključuje primjenu znanja, vještina, alata i tehnika u projektnim aktivnostima kako bi se ostvarili projektni zahtjevi. Projektni menadžment se ostvaruje primjenom i integracijom 47 logički grupiranih procesa projektnog menadžmenta koji su kategorizirani u pet procesnih grupa. Tih pet procesa su: iniciranje, planiranje, izvršavanje, kontrola i završavanje (Project Management Institute, 2013). Nadalje, PMI (2013) navodi kako upravljanje projektima uobičajeno uključuje:

- Analizu zahtjeva;
- Adresiranje raznolikih potreba, briga i očekivanja projektnih dionika u planiranju i izvršavanju projekta;
- Uspostavu, održavanje i izvršavanje komunikacije među projektnim dionicima;
- Vođenje dionika tijekom ispunjavanja projektnih zahtjeva i ostvarivanja ciljeva projekta;
- Održavanje ravnoteže među projektnim ograničenjima.

Sve navedene aktivnosti zapravo upućuju na to koja je uloga osobe zadužene za upravljanje projektom, odnosno projektnog menadžera, a upravo to će biti obrađeno u sljedećem poglavlju.

2.1.4. Uloga i odgovornosti projektnog menadžera

Projektni menadžer je osoba zadužena od strane organizacije/poduzeća da vodi tim koji je odgovoran za postizanje projektnih ciljeva. Ovisno o strukturi organizacije, projektni menadžer je odgovoran voditelju organizacijske jedinice[2] ili zajedno s ostalim projektnim menadžerima odgovara voditelju portfelja[3] ili osobi na sličnoj poziciji koja je u konačnici odgovorna za projekte širom poduzeća. Osim toga, projektni menadžer blisko surađuje i s drugim pozicijama poput poslovnog analitičara[4], menadžera za osiguranje kvalitete s ostalim stručnjacima iz različitih područja ekspertize. Generalno, projektni menadžeri imaju odgovornost udovoljiti potrebama zadataka, tima te individualnim potrebama. Budući da je to strateška disciplina, projektni menadžment postaje veza između strategije i tima. Projekti su nužni za rast i preživljavanje organizacija, oni kreiraju vrijednost u obliku poboljšanih poslovnih procesa, neophodni su u razvoju novih proizvoda i usluga te olakšavaju tvrtkama da

[2] eng. *functional manager*
[3] eng. *portfolio manager*
[4] eng. *business analyst*

odgovore na promjene u okruženju, konkurenciji i tržištu. Zbog toga uloga projektnog menadžera postaje sve više strateška (Project Management Institute, 2013).

Također, PMI (2013) navodi sljedeće interpersonalne vještine projektnih menadžera kao one od iznimne važnosti za njihov uspjeh: vodstvo, sposobnost izgradnje tima, motivacijske vještine, komunikacijske vještine, sposobnost utjecaja na druge, donošenje odluka, politička i kulturalna osviještenost, pregovaračke vještine, izgradnja povjerenja, upravljanje konfliktima te mentoriranje. Ovisno o grani industrije, projektni menadžeri će se koristiti specijaliziranim alatima i tehnikama te moraju imati specifično znanje, no navedene vještine su univerzalne vještine koje bi svaki projektni menadžer trebao posjedovati i aktivno raditi na njihovom unaprjeđenju. Razvijanje interpersonalnih vještina je uglavnom moguće samo kroz sami rad s ljudima pa se tako od projektnih menadžera često traži prethodno iskustvo rada u određenim poljima. To ujedno čini ulazne barijere za ovu poziciju iznimno rigoroznim te obično nije praksa da se osobe tek nakon završenog formalnog obrazovanja zapošljavaju na ovim pozicijama.

Ipak, u posljednje vrijeme možemo svjedočiti pojavi tzv. juniorskih pozicija za projektne menadžere. U tom slučaju, juniori asistiraju kolegama projektnim menadžerima u svakodnevnim aktivnostima, primjerice pisanju zapisnika nakon sastanka, dogovaranju sastanaka itd. Iako takva osoba možda nije zrela za samostalno vođenje projekata i odgovornosti koje idu s time, može se postepeno prilagođavati i učiti poslovnim procesima pojedinog poduzeća tj. organizacije te se tako pripremiti za ono što se od nje bude očekivalo kasnije.

2.1.5. Projektni tim

Projektni tim uključuje projektnog menadžera i grupu individualaca koji zajedno rade na izvršavanju projektnih ciljeva. Tim se sastoji od pojedinaca iz različitih grupa sa specifičnim znanjima i vještinama kojima mogu doprinijeti uspjehu projekta. Struktura i karakteristike projektnog tima mogu značajno varirati, ali jedna je konstantna uloga projektnog menadžera kao lidera tima, neovisno o tome koliki autoritet projektni menadžer ima nad članovima svojega tima (Project Management Institute, 2013).
Svakako da o tome koliki će autoritet projektni menadžer imati u projektnom timu ovisi ponajprije o ustrojstvu poduzeća tj. organizacije i prirodi projekta, međutim, projektni

menadžer je dio tima kao i svaki drugi član te ne bi trebao imati ulogu „nadređenoga“ ostalim projektnim članovima.

2.1.5.1. Projektni dionici

Prema Schwalbe (2015), projektni dionici su svi ljudi uključeni u projektne aktivnosti ili oni koji su direktno pod njihovim utjecajem. U dionike spadaju projektni sponzor, projektni tim, osoblje za podršku, kupci, korisnici, dobavljači te projektni protivnici. Budući da je spektar dionika veoma različit, različita su i njihova očekivanja od projekta. Kako bi se bliže prikazali pojedini dionici i njihove uloge, Schwalbe navodi primjer projekta izgradnje kuće:

- Sponzori projekta bi bili potencijalni kućevlasnici koji će jednom tu kuću platiti. S obzirom na to da će u većini slučajeva imati određeni budžet, očekivat će da će izvođači radova pružiti realnu procjenu kakvu kuću mogu dobiti s obzirom na financijske limite. Bez obzira na budžet, potencijalni kupci bi se trebali moći osloniti na to da će im izvođač radova pružiti donekle točnu procjenu svih troškova. Osim toga, trebaju i realnu procjenu kada bi mogli useliti u kuću. U ovom primjeru projektni sponzori su i kupci i korisnici krajnjeg proizvoda tj. kuće što u stvarnosti nije čest slučaj
- Kupnju kuće u većini slučajeva treba financirati kreditom banke ili druge financijske institucije koja će osigurati pravni interes na imanje. Ta institucija bi bila primjer pravnog dionika koji mora biti informiran o svim promjenama u planu i rasporedu provođenja projekta jer je projekt u tom slučaju dio pravnog ugovora.
- Projektni menadžer bi u ovom slučaju bio voditelj gradilišta tj. glavna osoba zadužena za sve aktivnosti koje se provode za vrijeme izgradnje. Projektni menadžer mora surađivati sa svim dionicima uključenima u projekt te je njegova odgovornost zadovoljiti njihove potrebe i očekivanja.
- Projektni tim za izgradnju kuće bi se sastojao od građevinskih radnika, električara, stolara itd.. Ovi dionici moraju znati što točno moraju raditi i kada. Osim toga, trebali bi znati hoće li im oprema biti osigurana i čekati ih na gradilištu ili su oni zaduženi za nabavku potrebnih materijala. Njihov rad treba biti koordiniran jer uključuje mnoge međusobno povezane faktore pa tako npr. stolar ne može krenuti ugrađivati kuhinjske elemente dok zidovi nisu postavljeni.

- Osoblje za podršku bi u ovom primjeru bio asistent/asistentica izvođaču radova. Ta osoba pruža podršku primjerice koordinirajući sastanke među ostalim dionicima.
- Dobavljači predstavljaju kanal za nabavu materijala kao što su drvo, stakla, podovi, kućni aparati i ostali materijali. Oni očekuju točne detalje u vezi proizvoda koje pružaju, kao i informacije gdje i kada ih treba dostaviti.
- Protivnici su osobe kojima provođenje projekta nije u interesu. U navedenom primjeru, to bi mogli biti primjerice susjedi kojima smeta buka za vrijeme izgradnje kuće. Važno je i njih uzeti u obzir jer svojim djelovanjem mogu otežati ili usporiti provođenje projekta.

Ovaj relativno jednostavan primjer je naveden kako bi se jasnije prikazale sve strane koje imaju određeni interes za vrijeme provođenja projekta. Slična pravila i uloge vrijede za više-manje sve tipove projekata (građevinske, informacijsko-komunikacijske, istraživačke itd.), a s veličinom obuhvata raste i broj uloga i osoba koje će na projektu sudjelovati. Projektni menadžeri moraju za čitav vijek trajanja projekta imati na umu sve interesne skupine te sa svima graditi što bolje odnose.

2.1.5.2. Područja znanja u projektnom menadžmentu

Područja znanja o upravljanju projektima opisuju ključne kompetencije koje projektni menadžeri moraju razviti. Schwalbe (2015) ih dijeli u deset ključnih područja, a to su:

1. Upravljanje obuhvatom projekta uključuje definiranje i upravljanje svog potrebnog posla da bi se projekt uspješno završio.
2. Upravljanje vremenom uključuje procjenjivanje koliko vremena će biti potrebno dok se projekt ne završi, razvoj primjerenog rasporeda i brigu o tome da se projekt dovrši unutar dogovorenog vremena.
3. Upravljanje troškovima se sastoji od kreiranja budžeta i upravljanja budžetom projekta.
4. Upravljanje kvalitetom projekta podrazumijeva to da projekt ispuni predefinirane zahtjeve za kvalitetom proizvoda/usluge koja je produkt projekta.
5. Upravljanje ljudskim resursima ima za cilj brigu o efikasnosti ljudi koji rade na projektu
6. Upravljanje komunikacijom na projektu podrazumijeva generiranje, prikupljanje, širenje i pohranjivanje informacija o projektu.

7. Upravljanje rizikom projekta uključuje identifikaciju, analizu i odgovore na rizike kojima je projekt podložan.
8. Upravljanje projektnom nabavom uključuje nabavu roba i usluga za projekt od trećih strana koje nisu direktno uključene u projekt.
9. Upravljanje dionicima projekta podrazumijeva identifikaciju i analizu potreba dionika te istovremeni nadzor njihovog angažmana na projektu.
10. Integracijsko upravljanje je opća funkcija koja utječe i na koju utječu sva druga područja znanja o upravljanju projektima.

Unatoč prednostima kojima projektni menadžment doprinosi projektu, on nije magična metoda koja garantira uspjeh na svim projektima. Projektni menadžment je dosta široka, a često i kompleksna disciplina. Zbog toga je važno da projektni menadžeri razvijaju svoja znanja i vještine kontinuirano te da uče iz vlastitih, ali i tuđih pogrešaka i uspjeha (Schwalbe 2015).

Uspješnost projektnih menadžera svakako će korelirati i s njihovim iskustvom te poznavanjem specifične industrije u kojoj djeluju. Iako su navedena područja znanja univerzalna za svaku industriju, posebna znanja dolazit će do izražaja ovisno o industriji i kompleksnosti projekata.

2.2. Važnost projektnog menadžmenta u razvoju softverskih proizvoda

Projektni menadžment se intenzivno koristi u industrijama informacijsko-komunikacijske tehnologije, građevinarstvu i vojnoj (obrambenoj) industriji. Tako se poduzeća koja se bave razvojem softverskih proizvoda sve više počinju oslanjati na projektni menadžment i ustaljenje inženjerske prakse da bi lansirali proizvode u današnjem iznimno kompetitivnom okruženju (Ebert, 2002).

Kao što je navedeno u općenitoj definiciji, projektni menadžment u razvoju softverskih proizvoda se bavi iniciranjem, planiranjem, praćenjem i kontrolom svih aktivnosti u procesu te informiranjem dionika o statusu napretka projekta. Proces ispravnog razvoja softvera treba slijediti osnove softverskog inženjerstva i uzeti u obzir analizu zahtjeva, funkcionalne i tehničke specifikacije, modele podataka, dokumentacijske standarde, testiranje, održavanje i osiguravanje kvalitete softvera. Kako bi se uspješno razvio krajnji proizvod, projektni menadžment i proces razvoja softvera trebaju biti integrirani.

2.2.1. Softverski inženjering

Softverski proizvod je skup računalnih programa i pripadne dokumentacije stvoren zato da bi se prodao. Može biti razvijen za određenog korisnika[5] ili općenito za tržište[6]. Softverski proizvod ćemo često kraće nazivati softver ili (softverski) sustav (Manger 2016).

Danas se od softvera očekuje da odlikuje sljedećim atributima kvalitete (Manger, 2016):

- Mogućnost održavanja. Softver se mora moći mijenjati u skladu s promjenjivim potrebama korisnika.
- Pouzdanost i sigurnost. Softver se mora ponašati na predvidiv način te ne smije izazivati fizičke ili ekonomske štete.
- Efikasnost. Softver mora imati zadovoljavajuće performanse te treba upravljati strojnim resursima na štedljiv način.
- Upotrebljivost. Softver treba raditi ono što korisnici od njega očekuju, sučelje mu treba biti zadovoljavajuće te za njega mora postojati dokumentacija.

Nadalje, prema Mangeru (2016), softversko inženjerstvo je znanstvena i stručna disciplina koja se bavi svim aspektima proizvodnje softvera. Dakle, softversko inženjerstvo se bavi modelima, metodama i alatima koju su potrebni da bi se na što kvalitetniji način mogli proizvoditi što kvalitetniji softverski proizvodi.
Strukturu softverskog inženjeringa čine tri osnovne komponente: metode, alati i postupci (procedure). Njihovo jedinstvo definira kvalitetu razvoja pa je zbog toga značajno da se odaberu one komponente koje se postavljenim zadacima i problemima u razvoju najlakše prilagođavaju (Sommerville, 1992 navedeno u Čubranić, Kaluža i Novak, 2013). Uz to, Čubranić, Kaluža i Novak (2013) iznose kako donošenje prave odluke nije jednostavan zadatak zbog individualnog i kreativnog karaktera postupka razvoja, zbog različitosti pojedinih sustava za koji se razvija softver i različitog okruženja sustava, ali i zbog nepostojanja „recepata“ za izbor.

S obzirom na kompleksnost softverskog inženjeringa i razvoja softverskih proizvoda, primjena projektnog menadžmenta je gotovo pa neophodna za svaku vrstu projekta koji ima barem minimalnu dozu kompleksnosti.

[5] eng. *bespoke product, customized product*
[6] eng. *generic product*

2.2.2. Kontekst projekata u industriji informacijsko-komunikacijske tehnologije

Za razliku od projekata u drugim industrijama, projekti u industriji informacijske-komunikacijske tehnologije[7] znaju biti veoma različiti. Neki uključuju manji broj ljudi te instalaciju pripadajućeg softvera na hardver, dok drugi znaju brojati na stotine ljudi koji analiziraju nekolicinu organizacijskih poslovnih procesa te zatim razvijaju novi softver u suradnji s korisnicima kako bi se zadovoljile poslovne potrebe. Spomenuti projekti služe i kao podrška u gotovo svim ostalim industrijama. Tako na primjer, upravljanje projektom za odjel animacija nekog filmskog poduzeća zahtjeva različite vještine od projekta poboljšanja sustava naplate poreza ili projekta instalacije komunikacijske infrastrukture u zemlji trećega svijeta. Zbog raznolikosti ICT projekata i zbog činjenice da je to polje relativno mlado, važno je razviti i slijediti najbolje prakse u upravljanju tim raznolikim projektima (Schwalbe, 2015).

2.2.3. Karakteristike timova u projektima razvoja softverskih proizvoda

Karakteristike ICT industrije i priroda poslova koje ljudi u njoj obnašaju diktiraju i izgradnju projektnih timova zaduženih za ostvarenje ciljeva projekta. „Recept“ koji funkcionira u jednoj grani djelatnosti, ne mora nužno važiti i za drugu pa se prije svega potrebno osvrnuti na „jezgru“ svakog projekta tj. projektni tim.

Zbog činjenice da su projekti u ICT industriji toliko različiti, ljudi uključeni u njih imaju različita zaleđa te posjeduju različite setove vještina. Zbog spomenute raznolikosti, ovi timovi imaju prednosti u analizi zahtjeva projekata jer ih mogu sagledati iz različitih točki gledanja. Mnoga poduzeća namjerno zapošljavaju diplomante koji su diplomirali na drugim poljima, kao što su neka od polja društvenih znanosti (pr. ekonomija), matematika i slično kako bi pružali bolju perspektivu. No, bez obzira na raznolikost u obrazovanju, postoje neke ustaljene pozicije koje su karakteristične za poslove u ovoj industriji, a to su: poslovni analitičar, sistemski i mrežni administrator, programer, analitičar baze podataka, specijalisti sigurnosti, softver testeri, hardverski specijalisti, softver inženjer, sistemski arhitekt itd.. Pod pojmom programera podrazumijevamo različite pozicije ovisno o tehnologiji kojom se osoba bavi, a neke od njih su Java, PHP, C/C++/C# (Schwalbe, 2015).

[7] eng. *Information Communication Industry*

S obzirom da se nove tehnologije pojavljuju neprestano, u skladu s njima se otvaraju i mijenjaju navedene pozicije. Nadalje Schwalbe (2015) iznosi kako neka poduzeća u ovoj industriji zahtijevaju specijalizirane vještine za samo neku od tehnologija, dok neke pozicije zahtijevaju poznavanje više tehnologija pr. sistemski arhitekt. Jednom kada krenu raditi na određenoj poziciji tj. u određenoj tehnologiji, radnici se uglavnom specijaliziraju u tom polju ili eventualno naprave prijelaz u neku od menadžerskih pozicija. Također, velik broj projekata uključuje rad radnika na određeno vrijeme. Ti radnici nisu stalni zaposlenici poduzeća koje provodi projekt, već su unajmljeni od treće strane kako bi obavili neki specifičan zadatak. Rad s tzv. „slobodnim agentima“ Schwalbe (2015) smatra posebnim izazovom u vođenju projekata.

Kao što je već i spomenuto, mnogi poslovi, odnosno pozicije u industriji informacijsko-komunikacijskih tehnologija reflektiraju različite tehnologije koje je potrebno poznavati za svaku pojedinu poziciju. S obzirom na raznolikost u tehničkom znanju, postoji potencijalna komunikacijska barijera koja može uzrokovati znatne probleme. Dizajner korisničkih sučelja[8] možda neće razumjeti jezik analitičara baze podataka i obrnuto.

To ne mora biti slučaj samo za odvojene pozicije već postoji opasnost od komunikacijskih barijera unutar iste pozicije npr. programeri koji rade u različitim tehnologijama. Te okolnosti doprinose izazovima s kojima se susreću projektni menadžeri pri vođenju projekata ove vrste. Osim raznolikosti tehnologija, one se i konstantno mijenjaju. Projektni tim se može naći u situaciji u kojoj su blizu završetka projekta otkrili tehnologiju koja značajno može pospješiti projekt ili neku od dugoročnih projektnih potreba. Zbog te činjenice se vremenski limit za izvođenje projekta poprilično smanjio. Visoko dinamična okolina je stoga dovela do potrebe za uvođenjem visoko dinamičnih procesa kako bi se proizveli ciljani proizvodi (Schwalbe, 2015).

Ti visoko dinamični procesi rezultirali su pojavom novih trendova u vođenju razvoja softverskih proizvoda koji su zbog benefita koje imaju na uspjeh projekata postali ustaljena praksa u većini projekata u informacijsko-komunikacijskoj industriji. Upravo je ta dinamičnost dovela do novih oblika suradnje i rada u projektnim timovima koja je do nedavno bila nezamisliva.

[8] eng. *user interface*

2.2.4. Novi trendovi u vođenju razvoja softverskih proizvoda

Nedavni trendovi kao što je globalizacija, outsourcing, virtualni timovi i agilno vođenje projekata kreiraju nove izazove, ali i prilike za projektne menadžere i njihove timove. Ovo poglavlje će se dotaknuti globalizacije, outsourcinga i virtualnih timova dok će agilno vođenje projekata biti razrađeno u sklopu četvrtog poglavlja rada.

2.2.4.1. Globalizacija

Friedman opisuje efekte globalizacije, koja je stvorila „ravnu“ zemlju u kojoj su svi povezani, a igralište[9] je za sve jednako. Niže trgovinske i političke barijere i digitalna revolucija su omogućile instantnu interakciju milijardi ljudi diljem planete. Za individualce i mala poduzeća to znači da napokon imaju priliku natjecati se s velikim korporacijama. Friedman također priča o porastu učitavanja sadržaja na Internet pomoću kojeg ljudi mogu dijeliti znanje putem internetskih blogova, podcasta i softvera otvorenog koda. Informacijska tehnologija je prema Schwalbe ključni omogućivač globalizacije (Schwalbe, 2015).

Tako je primjerice u lipnju 2019. godine Facebook brojio 2.41 milijardu korisnika, od kojih je 1.59 milijardi dnevno aktivno, što znači da barem jednom u danu posjete Facebook stranice (Noyes, 2019).

To su brojke koje bi bile neostvarive da globalizacija nije omogućila korporacijama istovremenu prisutnost na gotovo svim tržištima diljem svijeta te time zauvijek promijenila poslovne modele poduzeća u nastajanju. Iako je većina velikih poduzeća kao što su Apple, Facebook, Google započela s poslovanjem u Sjedinjenim Američkim Državama, danas im je tržište čitav svijet. S obzirom na razvijenost internetske infrastrukture, danas više nije potrebno fizički biti prisutan na određenom tržištu kako bi se na njemu prodavao proizvod ili usluge već je pojedincima i poduzećima za početak potrebno samo osobno računalo i internetska veza.

Prilikom rada na globalnim projektima, Schwalbe (2015) naglašava sljedeće barijere s kojima se susreću projektni menadžeri:

[9] eng. *playing field*

- Komunikacija: Zbog činjenice da ljudi rade u različitim vremenskim zonama, govore različitim jezicima, imaju drugačije kulturološko zaleđe te slave različite blagdane, potrebno je adresirati kako će oni komunicirati na učinkovit i pravovremen način. Zato je plan upravljanja komunikacijama od vitalne važnosti.
- Povjerenje: Povjerenje je bitan aspekt svakog tima, a pogotovo onih kojima su članovi fizički razdvojeni. Potrebno je od samog početka graditi povjerenje prepoznavajući i poštivajući tuđe razlike i vrijednosti koje doprinose projektu.
- Uobičajene radne prakse: Važno je uskladiti radne procese i razvijati pravila rada koja svima odgovaraju i s kojima se svi slažu. Projektni menadžeri moraju timovima dozvoliti da sami razviju dio praksi, a tome im mogu pomoći neki od alata koji su prethodno navedeni.
- Alati: Informacijsko-komunikacijska tehnologija ima vitalnu ulogu u globalizaciji, pogotovo u pospješivanju komunikacije i radnih praksi među timovima. Mnogi za komunikaciju koriste besplatne alate kao što su Skype, Google Dokumenti ili društvene mreže. Mnogi softverski alati za projektni menadžment sadrže vlastite komunikacijske i kolaboracijske funkcionalnosti u integriranom paketu. IBM nastavlja biti lider u pružanju takvih kolaboracijskih alata poduzećima u preko 175 zemalja svijeta. Prati ga Oracle u 145 zemalja, SAP u 130 te Microsoft u 113 (IBM 2014 navedeno u Schwalbe, 2015).

2.2.4.2. Outsouricng

Outsourcing često podsjeća na vanjsko ugovaranje i druge poslovne odnose s partnerskim poduzećima, primjerice dobavljačima, no detaljnija definicija govori kako se outsourcingom određeni poslovni procesi prepuštaju specijaliziranim partnerskim poduzećima koja imaju mogućnost te poslovne procese obavljati kvalitetnije i jeftinije, uz zadržavanje komunikacije s poduzećem s kojim su potpisali ugovor o outsourcingu. Na taj se način poduzeće koje je odlučilo izdvojiti određene poslovne procese može usmjeriti na ključne poslovne procese (Latinović, 2010 navedeno u Liović, 2016).

Zbog sve većeg korištenja outsourcinga u projektima, projektni menadžeri moraju biti što bolje upoznati s mnogim globalnim pitanjima, uključujući rad i upravljanje virtualnim timovima.

2.2.4.3. Virtualni timovi

Nekoliko faktora je pridonijelo razvoju tzv. virtualnih timova, a neki od njih su: vrijeme i trošak putovanja ili realokacije radnika, mogućnost rada na daljinu i smanjivanje komunikacijskih barijera, prednosti zapošljavanja radnika na lokacijama s manjim troškovima života, preference radnika po pitanju fleksibilnosti radnih sati itd..Virtualni tim je grupa ljudi koji rade zajedno bez obzira na vremenska i prostorna ograničenja, koristeći komunikacijske tehnologije (Schwalbe, 2015). Schwalbe smatra da su glavne prednosti i nedostaci virtualnih timova sljedeći:

Prednosti:

- Smanjeni troškovi jer većina radnika koji rade na daljinu ne treba uredski prostor niti infrastrukturu van one koje imaju u svojim domovima.
- Povećana konkurentnost zbog veće ekspertize radnika te činjenice da zbog toga što članovi rade diljem svijeta, mogu raditi u bilo koje doba dana i noći.
- Micanje barijera između poslovnog i privatnog života eliminiranjem fiksnih uredskih sati i potrebe za putovanjem na posao.

Nedostaci:

- Izolacija članova tima koji se ne prilagode dobro radu u virtualnom okruženju.
- Potencijalni komunikacijski problemi zbog odsustva neverbalne komunikacije pr. otežano stvaranje povjerenja među članovima.
- Smanjena mogućnost umrežavanja među članovima tima i prijenosa informacija neformalnim putem.
- Povećana ovisnost o tehnologiji kako bi se ostvarili poslovni ciljevi.

Nekoliko istraživanja je pokušalo odrediti čimbenike koji su u pozitivnoj vezi s učinkovitošću virtualnih timova. Ona uglavnom sugeriraju da timski procesi, odnosi temeljeni na povjerenju, stil vođenja i odabir članova tima pružaju najčvršće korelacije s performansama tima i zadovoljstvom članova (Schwalbe, 2015).

Osim fizičke prisutnosti i stvarnog doticaja s kolegama, danas gotovo da ne postoje druge barijere u radu na daljinu. Tako je u posljednje vrijeme činjenica da u poduzeću rade zaposlenici koji su zaposleni isključivo na daljinu postala nešto čime se poduzeća ponose.
Samo jedno od mnogih je poduzeće Zapier koje se bavi automatizacijom aplikacija kako bi korisnici mogli integrirati svoja softverska rješenja s tisućama drugih, a partneri su im tehnološki giganti kao što su Google, Salesforce, Dropbox itd.. Zapierovi zaposlenici rade u 17 vremenskih zona i 24 države, a jedini uvjet za takav rad je stabilna internetska veza. Njihovi radnici komuniciraju asinkrono, rade autonomno te se ponose svojim poslom (Zapier, 2019).

2.3. Izazovi projektnog menadžmenta u razvoju softverskih proizvoda

Za prikaz najvećih izazova u vođenju projekata razvoja softverskih proizvoda koristit će se anketa provedena u sklopu istraživanja izazova u upravljanju projektima razvoja softverskih proizvoda.
Prema Demiru (2009), većina publikacija na temu izazova u vođenju projekata razvoja softvera je bazirano na iskustvima istraživatelja, a ne na empirijskim istraživanjima. Njegov primarni cilj je bio identificirati izazove s kojima se susreću projektni menadžeri u projektima razvoja softverskih proizvoda. To istraživanje je različito od ostalih po tome što ima širi pogled, a fokusira se na područja menadžmenta više nego na specifične razloge zašto do izazova dolazi. U istraživanju je sudjelovalo 78 sudionika na projektima razvoja softverskih proizvoda diljem svijeta, od menadžera na različitim razinama do članova razvojnih timova. Odgovori su dolazili od Sjeverne Amerike, Europe, Azije do Južne Amerike. Većina odgovara je iz Sjeverne Amerike u kojoj se i producira značajan dio svjetskog softvera. U prvom dijelu ankete prikupljali su se podaci o pozadini sudionika tj. o njihovom iskustvu na različitim pozicijama u razvoju softvera, uključujući i menadžerske pozicije. Drugo, bili su ispitivani o broju projekata na kojem su sudjelovali. 42 ispitanika od ukupnih 78 je sudjelovalo u manje od 15 projekata. 36 od 78 sudionika je sudjelovalo u više od 78 projekata. U tablici 2 je prikazano iskustvo sudionika u terminima pozicija koje su za života obnašali u razvoju softvera. Tablica uključuje sve uloge koje je pojedini sudionik imao tijekom svoje karijere.

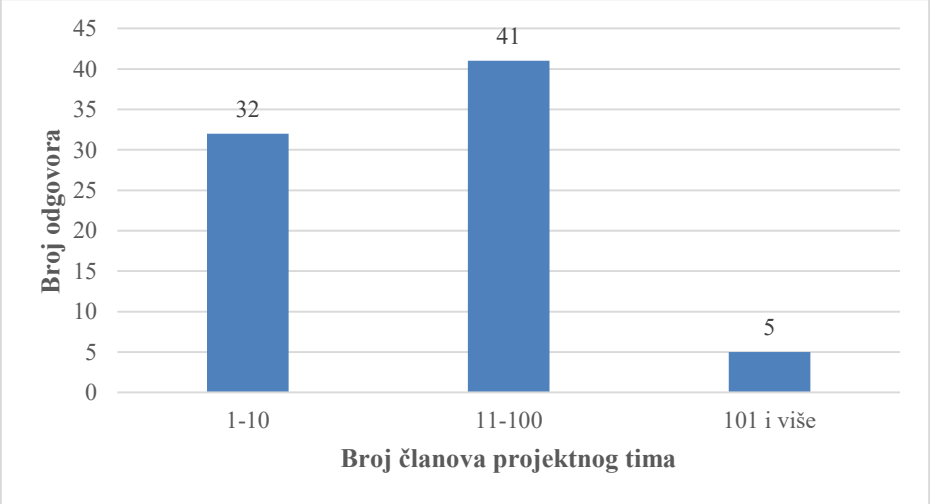
Tablica 1 Iskustvo sudionika u terminima pozicija na kojima su radili za vrijeme karijere

Pozicija	**Broj odgovora**
Projektni menadžer	56
Projektni voditelj tima	49
Inženjer zahtjeva[10]	23
Softver Arhitekt	27
Softver Dizajner	24
Softver Tester	15
Održavaoc/Održavatelj softvera	18
Programer	43
Istraživač/Znanstvenik	28
Sistemski inženjer	24
Drugo	26
Ukupan broj ispitanika	**78**

Izvor: Demir (2009).

Nadalje, odgovori na pitanja ispitanika su prikazani u grafikonima 1, 2, 3 i 4.

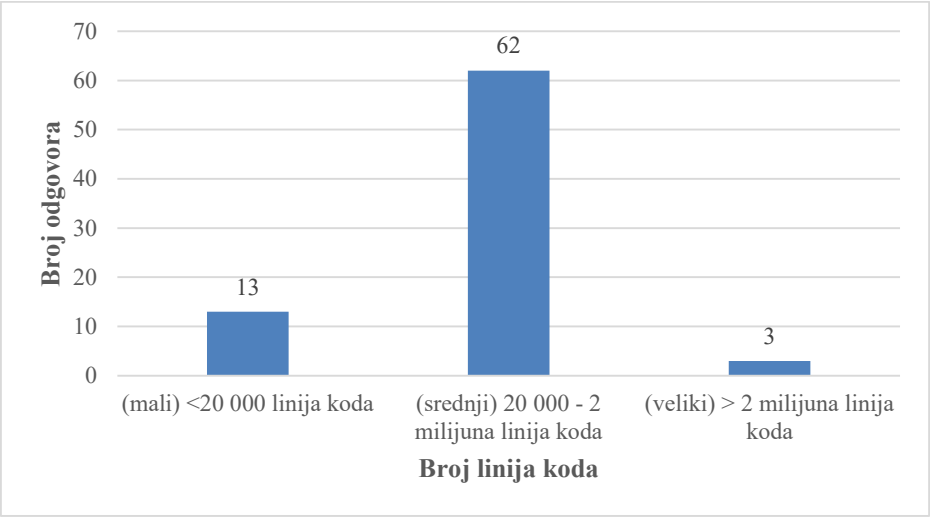
Grafikon 1 Veličina projekta po broju članova



Izvor: Demir (2009).

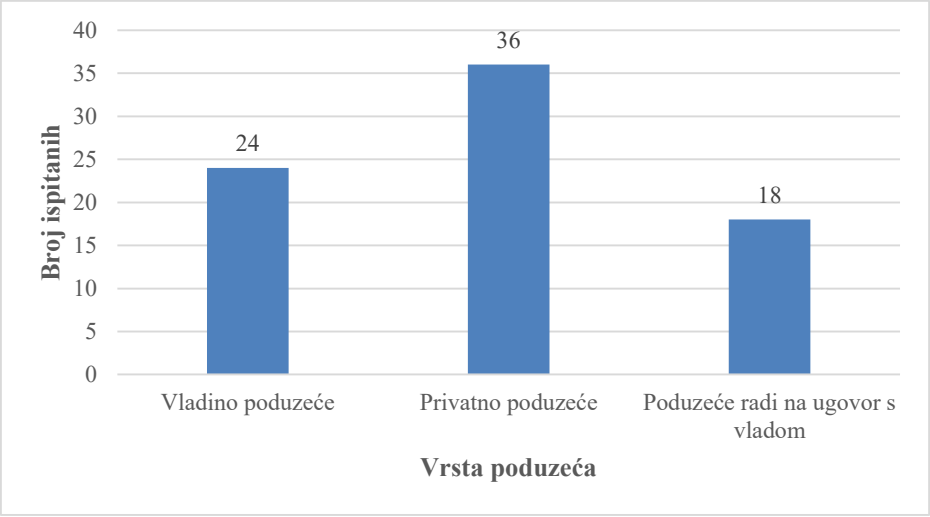
[10] eng. *Requiremets Engineer*

Grafikon 2 Veličina projekta po broju linija koda



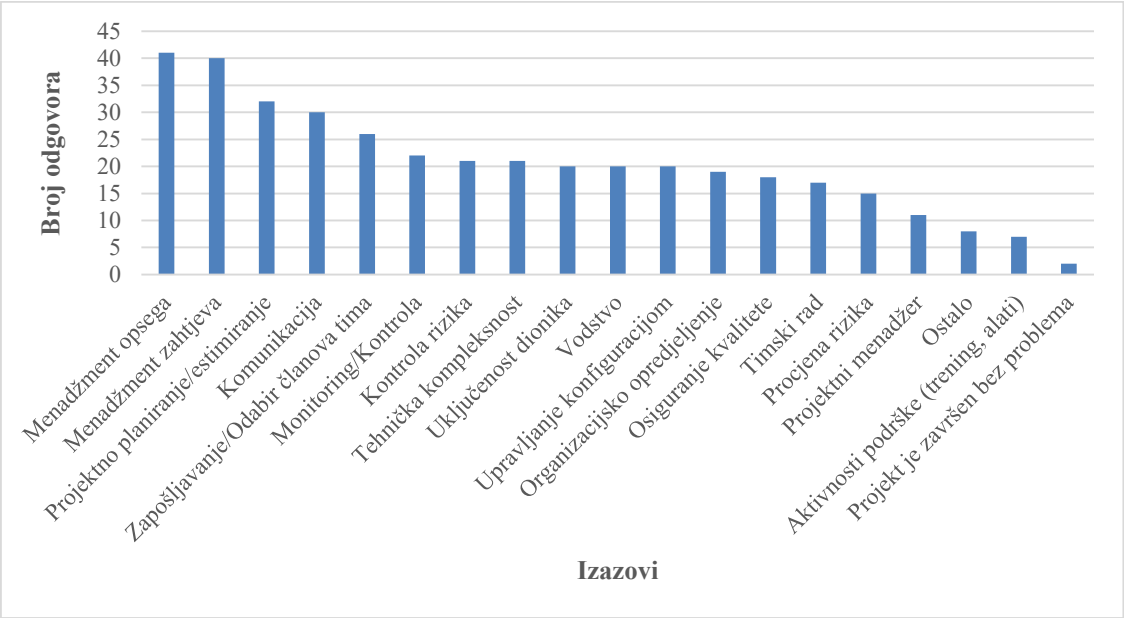
Izvor: Demir (2009).

Grafikon 3 Vrsta poduzeća u kojem se projekt odvijao



Izvor: Demir (2009).

Grafikon 4 Izazovi s kojima su se ispitanici susreli na posljednjem projektu



Izvor: Demir (2009).

Prepoznati je li na projektima bilo izazova ili ne je poprilično težak zadatak. U tablici 3 su navedeni relativni iznosi dobiveni iz rezultata pitanja prikazanog na grafikonu 4. Iz nje je najlakše očitati koje su stavke smatrane najvećim izazovima prilikom rada na projektima. Ispod nje će biti pobliže opisano pet područja s najvišim postotcima odgovora, odnosno pet područja u kojima se više od trećine ispitanika susrelo s izazovima na posljednjem projektu na kojem su radili.

Tablica 2 Područja izazova u projektima razvoja softverskih proizvoda

Područja projektnog menadžmenta	**Postotak odgovora**	**Broj odgovora**
Menadžment obuhvata	52.6%	41
Menadžment zahtjeva	51.3%	40
Projektno planiranje/procjenjivanje	41.0%	32
Komunikacija	38.5%	30
Zapošljavanje/odabir članova tima	33.3%	26

Monitoring/kontrola	28.2%	22
Kontrola rizika	26.9%	21
Tehnička kompleksnost	26.9%	21
Uključenost dionika	25.6%	20
Vodstvo	25.6%	20
Upravljanje konfiguracijom	25.6%	20
Organizacijsko opredjeljenje	24.4%	19
Osiguranje kvalitete	23.1%	18
Timski rad	21.8%	17
Procjena rizika	19.2%	15
Projektni menadžer	14.1%	11
Ostalo	10.3%	8
Aktivnosti podrške	9.0%	7
Projekt je završen bez problema	2.6%	2

Izvor: Demir (2009).

Observacije Demira (2009) vezane uz područja s najvišim postotkom odgovora su sljedeće:

- Menadžment obuhvata: U svim kategorijama projekata, menadžment obuhvata je viđen kao jedan od top tri izazova na projektima. Omjeri projekata koji su se susreli s ovim izazovom varira od 61.5% do 41.7%. Ukratko, gotovo svaki drugi projekt se susreo s izazovom upravljanja obuhvatom.
- Menadžment zahtjeva: U svim kategorijama projekata, menadžment zahtjeva je viđen kao jedan od top dva izazovna područja. Omjeri variraju od 61.1% do 45.8%. Opet, gotovo svaki drugi projekt se susreo s ovim izazovom.
- Projektno planiranje i procjenjivanje: Ovo područje projektnog menadžmenta se također nalazi u top 3 najizazovnijih. Vrijedi napomenuti da je naveden kao izazov

broj jedan u vladinim projektima, dok je gotovo pola ispitnika izjavilo da se susrelo s izazovima u tom području na zadnjem projektu.

- Komunikacija: Komunikacija, koja generalno nije pokrivena u većini istraživanja se također nalazi u samom vrhu. Česta je pretpostavka da se komunikacija ne nalazi na listi izazovnih područja u manjim projektima, ali rezultati pokazuju da gotovo pa i ne postoji razlika u omjerima među projektima s 1-10 članova i projekata s 11-100 članova. Također, rezultati pokazuju da komunikacija predstavlja izazov neovisno o vrsti organizacije. Ipak, donekle je veći problem kod vladinih organizacija u usporedbi s ostalim vrstama.
- Zapošljavanje/odabir članova tima: Ovo područje je također poprilično zanemareno u drugim istraživanjima. Kod svih kategorija projekata, zapošljavanje tj. odabir članova tima je u top sedam izazovnih područja. Zapošljavanje je povezano s brojem slobodnih ljudi s potrebnim vještinama u određenom polju. Demir vjeruje da su projektni menadžeri i poduzeća ograničeni u pružanju rješenja za izazove u ovom području, a ovo se pitanje proteže i na obrazovanje i obuku u softverskom inženjerstvu.

Iz dobivenih rezultata je jasno zaključiti da će se vještine i znanje u vođenju i postavljanju projekta pokazati ključnima u eliminiranju potencijalnih rizika i izazova. Zanimljivo je da je tehnička kompleksnost projekta tek osma po redu, promatrajući postotke odgovora, dok su se ispred nje našle prethodno spomenute aktivnosti. Ono što se i moglo pretpostaviti je da je gotovo pa i nemoguće da se tijekom trajanja projekta ne pojave nikakvi izazovi pa je samo 2,6% ispitanika odgovorilo tim odgovorom.

3. VODOPADNI MODEL KAO NAJKORIŠTENIJI TRADICIONALNI MODEL VOĐENJA RAZVOJA SOFTVERSKIH PROIZVODA

U trećem poglavlju rada, navesti će se i opisati modeli proizašli iz smjernica tradicionalne metodologije razvoja softverskih proizvoda, a posebno će se izdvojiti i opisati najpopularniji, vodopadni model. Iako su se karakteristike modernih softverskih proizvoda značajno promijenile, zanimljivo je kako zakonitosti ovih modela još uvijek pronalaze svoju primjenu u modernom razvoju softvera.

3.1. Definicija i modeli tradicionalne projektne metodologije

Tradicionalna metodologija je bazirana na uzastopnim serijama koraka u životnom ciklusu projekta kao što su definicija zahtjeva, kreiranje rješenja, razvijanje i testiranje.
Tradicionalna metodologija je ovisna o nizu predeterminiranih procesa i uzastopnom dokumentiranju koje se vrši kako projekt napreduje iz faze u fazu (Nifkorova, Nikulsins, Sukovskis, 2009 navedeno u Leau et al., 2012). Uspjeh projekta koji je razvijan tradicionalnom metodologijom ovisi o poznavanju svih zahtjeva prije nego što se krene s razvojem. To znači da implementiranje bilo kakvih promjena za vrijeme trajanja projekta može prouzrokovati dosta problema. S druge strane, olakšava definiranje troškova, kreiranje rasporeda i primjerenu alokaciju resursa (Leau et al.,2012).

Tipičan primjer spomenutih faza prikazan je na slici 1 koja prikazuje isporuku projekata primjenom nekog od modela tradicionalne metodologije.

Slika 1 Isporuka projekata prema tradicionalnoj metodologiji

Izvor: Anonymous (2012) navedeno u Javanmard i Alian (2015).

- **Inicijacija**: Podrazumjeva procjenu obuhvata sistemskih zahtjeva i obuhvata cjelokupnog projekta. Inicijacija bi trebala rezultirati funkcijskom specifikacijom u kojoj su sadržane sve informacije o karakteristikama budućeg proizvoda.
- **Planiranje (arhitektura i dizajn)**: Uključuje razvijanje razumijevanja rješenja s tehničke perspektive, kreiranje dizajna modularnih komponenti i njihovih interakcija i postavljanje standarda za rješavanje zajedničkih tehničkih pitanja.

  U ovoj fazi se tehnička infrastruktura budućeg softvera predstavlja u formi dijagrama i modela. Ovakav dizajn može izvući na površinu potencijalne probleme koji bi proizišli razvojem. Također, iz dizajna, programeri koji će kasnije razvijati proizvod prvi puta mogu dobiti jasniju sliku kojim putem će razvoj ići (Leau et al.,2012).

- **Izvršavanje (razvoj)**: Izrada koda u okolini specifičnoj za kulturu projekta. Zadaci se dodjeljuju prema individualnim vještinama, a razvoj se nastavlja sve do ciljeva ili prekretnica koji su postavljeni. Laue et al. (2012) navode kako se često razvoj raščlanjuje na manje zadatke koji se raspodjeljuju među raznim timovima ovisno o njihovim vještinama.

  Primjerice, u razvoju mobilnih aplikacija bi se razvoj trebao raščlaniti po platformama tj. odvojeno bi se razvijala aplikacija za platformu Android te aplikacija za platformu iOS. Već u fazi planiranja je potrebno prikladno prilagoditi zadatke karakteristikama i razlikama među platformama jer često različiti timovi ne mogu pratiti u potpunosti iste upute.
- **Kontrola**: Testiranje individualnih komponenti bi se trebalo odvijati paralelno s razvojem, a testiranje na razini čitavog sustava se vrši pred kraj projekta. Idealno, na kraju se uključuje i testiranje nad potencijalnim kupcima kako bi se potvrdilo da su zahtjevi ispunjeni ili kako bi se identificiralo promjene koje je potrebno implementirati (Javanmard i Alian, 2015).

S obzirom na navedeni niz aktivnosti kojeg prati tradicionalna metodologija, životni ciklus razvoja proizvoda će poprimiti određene karakteristike s obzirom na karakteristike procesa. Glavne karakteristike životnog ciklusa razvoja softverskih proizvoda prema tradicionalnoj metodologiji su (Devi, 2013 navedeno u Javanmard i Alian, 2015):

- Cilj je temeljito razumjeti potrebe korisnika, oblikovati dizajn, besprijekorno razviti proizvod i implementirati ga kao funkcionalni sustav koji zadovoljava potrebe korisnika.
- Velik je naglasak na temeljitom planiranju kako bi se moglo nositi s rizicima.
- Temelje se na odabiru najboljeg načina za dostizanje željenog stanja iz početnog stanja., ali i na identificiranju alternativnih načina.
- Pristup pretpostavlja da su problemi dobro definirani i da se optimalno rješenje može postići opsežnim planiranjem unaprijed.
- Također pretpostavlja da su procesi predvidljivi, da se mogu optimizirati i standardizirati tj. učiniti lako ponovljivima.
- Pretpostavka je da se procesi mogu adekvatno mjeriti te da se izvori varijacija mogu prepoznati i kontrolirati tijekom životnog ciklusa razvoja.
- Orijentirane su prema procesima.

Ovisno o spomenutim karakteristikama razvoja proizvoda, poduzeća prihvaćaju stil vođenja koji je temeljen na naredbama i kontroli uz pretpostavku da ima utemeljenu hijerarhiju. Zbog toga su to pretežito mehanicistička[11] poduzeća koja imaju za cilj visoke performanse u stabilnom okruženju. Takva poduzeća karakterizira visoka formalizacija i standardizacija. Radnici s različitim specijalizacijama imaju uloge u postizanju definiranih rezultata ovisne o specijalizaciji. Također, proizvode se značajne količine dokumentacije koja objašnjava softver i opisuje tehničke i projektne specifikacije. Kupci imaju značajnu ulogu, ali je njihovo sudjelovanje maksimalno samo tijekom faza specifikacije i testiranja (Devi, 2013 navedeno u Javanmard i Alian, 2015).

Ukratko, postoje dva široka polja koja se javljaju u tradicionalnoj metodologiji oko kojih su problemi usredotočeni (Devi, 2013 navedeno u Javanmard i Alian, 2015):

- Životni ciklus razvoja i njegovi procesi: Način na koji su procesi shvaćeni, definirani te kako je njima upravljano (definiranost, predvidljivost, ponovljivost itd.) je procesno orijentiran te je fokus na postizanju stabilnosti.
- Stil vođenja: Način na koji su ljudi organizirani, vođeni i kontrolirani je kroz formaliziran i standardiziran pristup visokih zapovijedi i kontrole. Interakcija s korisnicima je u ovakvom stilu ograničena.

Tradicionalne, planski-orijentirane metodologije su dobar odabir za manje, dobro definirane projekte s ograničenim obuhvatom posla i nekolicinom varijabli. Imajući na umu rapidan rast ICT industrije, metodologije su morale evoluirati da bi se skuplji projekti s multimilijunskim budžetima mogli završiti na vrijeme i unutar budžeta (Anonymous, 2012 navedeno u Javanmard i Alian, 2015).

Modeli životnog ciklusa razvoja softvera[12] se općenito koriste za opisivanje koraka u okviru životnog ciklusa proizvoda. Potrebno je imati na umu da se model razlikuje od metodologije u smislu da model opisuje što točno treba činiti, dok metodologija opisuje kako to treba činiti. Dakle model je opisujući pojam, dok je metodologija propisujuća (Ruparelia, 2010). U nastavku će se pobliže opisati osnovni modeli koji slijede propise tradicionalne metodologije.

3.1.1.1. Vodopadni model

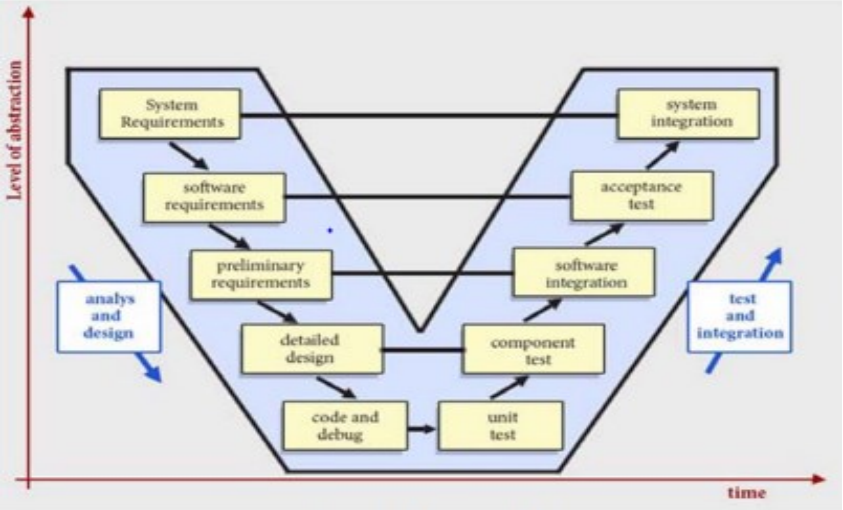
Vodopadni model naglašava strukturirani napredak između definiranih faza. Svaka se faza sastoji od definiranog seta aktivnosti i rezultata koji moraju biti izvršeni prije nego što započne sljedeća faza. Faze se često različito nazivaju, ali je osnovna ideja da prva faza obuhvaća ono što će sustav raditi sa svojim sustavnim i softverskim zahtjevima. Druga faza određuje kako će proizvod biti dizajniran. Treća faza je ona u kojoj programeri počinju pisati kod. Četvrta faza je testiranje sustava, a peta faza je fokusirana za zadaće provedbe kao što su obuka ljudi koji će koristiti proizvod te izrada sve potrebne dokumentacije. Jako je bitno naglasiti da se u inženjerskoj praksi izraz vodopadni model[13] koristi kao generičko ime za sve sekvencijalne modele softverskog inženjerstva (Awad, 2005).
Osim toga, u kolokvijalnom govoru se često tradicionalna metodologija poistovećuje s vodopadnim modelom upravo zbog popularnosti vodopadnog modela što je, dakako, pogrešno. Potrebno je razlikovati metodologiju od modela kako bi se mogle razumjeti njihove svrhe.

[12] eng. *software development lifecycle models*
[13] eng. *waterfall model*

3.1.1.2. V-model

V-model se može smatrati ekstenzijom vodopadnog modela, ali umjesto linearnog tijeka aktivnosti, u ovom modelu nakon faze programiranja tijek dobiva smjer „prema gore“ kao što je prikazano na slici 2.

Slika 2 V-model



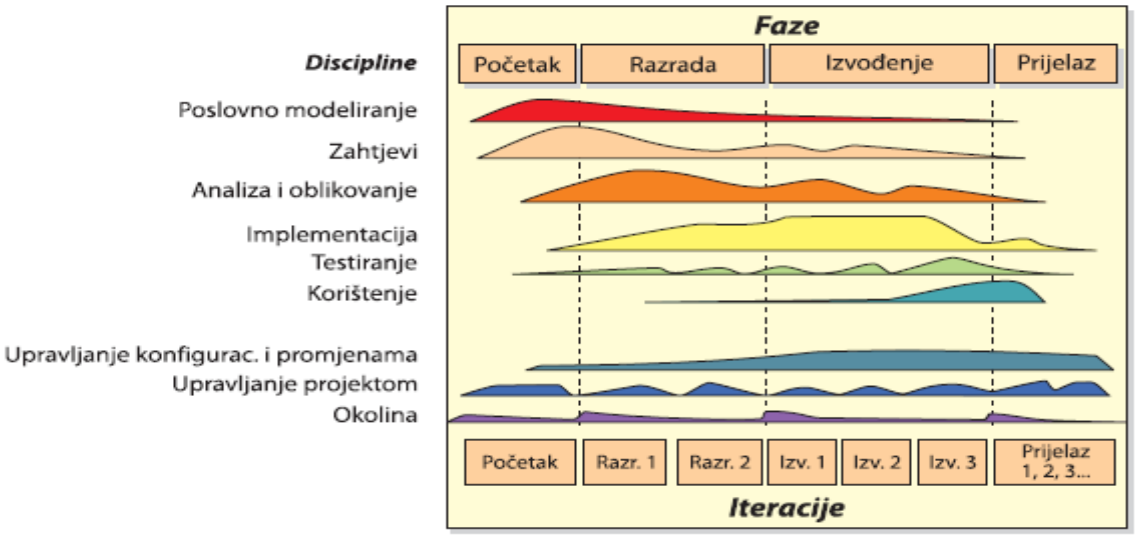
Izvor: Kumar i Bhatia (2014).

V-model uspostavlja odnos između svake faze razvoja u životnom ciklusu i s njima povezanim fazama testiranja. S obzirom na to da se testni scenariji razvijaju od početka i provode kontinuirano, ovaj model ima veće šanse za uspjeh od vodopadnog (Nabil, 2010 navedeno u Kumar i Bhatia, 2014). No, Kumar i Bhatia (2014) navode kako ni ovaj model ne pruža jasan put za rješavanje problema ako se suoči s njima u fazama testiranja.
Činjenica je da se faze testiranja povezuju s ostalim fazama u životnom ciklusu što je svakako napredak u odnosu na vodopadni model. No upitno je koliko je vremena ostavljeno za stvarni popravak problema koji se pojave tijekom testiranja. U tom slučau, alokacija vremena za testiranje ima presudnu ulogu za uspjeh projekta.

3.1.1.3. Model unificiranog procesa razvoja

U modelu unificiranog procesa razvoja[14] se svi napori, uključujući i modeliranje, organiziraju u radne tijekove[15]. U unificiranom procesu se oni izvode iterativno i inkrementalno. Životni ciklus unificiranog procesa prikazan je na slici 3.

Slika 3 Životni ciklus u unificiranom procesu razvoja



Izvor: Panian i Ćurko (2010).

Unificirani proces razvoja koristi objektno usmjerenu arhitekturu pomoću koje se mogu kreirati sustavi koji su lako proširivi, višekratno iskoristivi te intuitivno razumljivi. Također, koristi softver za vizualno modeliranje pomoću kojega je kod moguće prikazati kao dijagram notacija koji pak dopušta tehnički slabije kompetentnim pojedincima bolje razumijevanje sustava. Tako i osobe slabijeg tehničkog znanja mogu značajnije doprinijeti razvoju. Zahtjevima se upravlja koristeći slučajeve uporabe[16] i scenarije koji su se pokazali iznimno efikasnim za udovoljavanje funkcionalnih zahtjeva kao i anticipiranju očekivanog ponašanja sustava. Dizajn je također iterativan i inkrementalan što pridonosi smanjenju rizika projekta, omogućava više povratnih informacija od kupaca te olakšava programerima da ostanu usredotočeni. Unificirani proces pomaže u planiranju kontrole kvalitete i osiguranju kvalitete u svim fazama projekta te u njih uključuje sve članove tima (Awad, 2005).

[14] eng. *Rational Unified Process*
[15] eng. *workflow*
[16] eng. *use case*

Sličnost s vodopadnim modelom je ta što i unificirani proces razvoja ima fiksno definirane faze, a glavna razlika je ta što te faze uključuju nekoliko iteracija. Te iteracije postat će temelj na kojem će se kasnije razvijati modeli agilne metodologije.

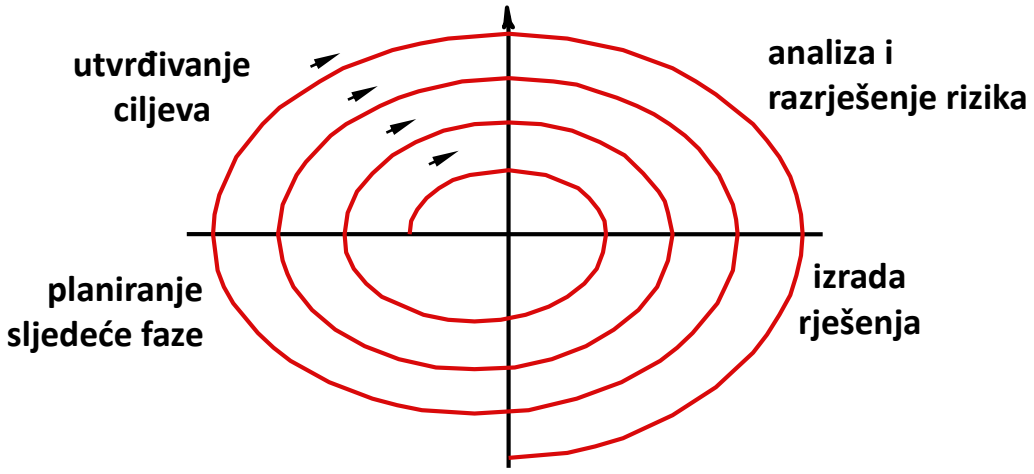
3.1.1.4. Spiralni model

Spiralni model kombinira elemente dizajna i prototipiranja u fazama s ciljem izvlačenja najboljih praksi iz koncepata implementacija od vrha prema dnu[17] ili pak od dna prema vrhu[18]. Spiralni model je prvi definirao Barry Boehm, nakon iskustva s višekratnim usavršavanjem vodopadnog modela kako bi ga prilagodio većim projektima razvoja softvera (Awad, 2005).

Faze u spiralnom modelu razvoja, prikazane na slici 4, su sljedeće:

1. Utvrđivanje ciljeva: specificiraju se ciljevi za svaku projektnu fazu
2. Analiza i razrješenje rizika: identificiraju se i analiziraju ključni rizici te se izvlače informacije o tome kako je rizik moguće smanjiti
3. Izrada rješenja: odabire se prikladan model za sljedeću fazu razvoja
4. Planiranje sljedeće faze: radi se osvrt na napravljeno nakon kojeg slijedi izrada planova za naredni krug spirale

Slika 4 Spiralni model



Izvor: Panian i Ćurko (2010).

[17] eng. *top-down*
[18] eng. *bottom-up*

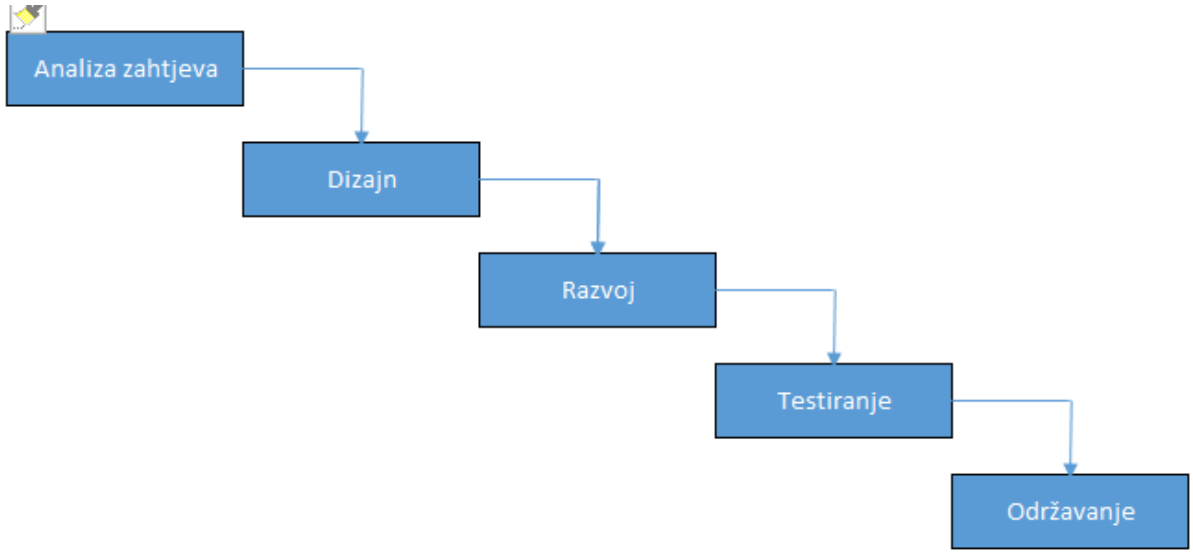
Iako postoji još varijacija koje su proizašle iz vodopadnog modela, najviše s ciljem prilagođavanja većim i kompleksnijim projektima, vodopadni model se ipak nametnuo kao najkorišteniji te stoga i najpopularniji model unutar tradicionalne metodologije.

3.2. Definicija vodopadnog modela

Vodopadni model životnog ciklusa softvera sekvencijalni je razvojni proces u kojem se smatra da napredak „teče prema dolje“ (slično vodopadu) kroz niz faza koje se moraju izvršiti da bi se uspješno napravio računalni softver (Bassil, 2012). Originalno, vodopadni model je predstavio Winston W. Royce 1970 godine kako bi opisao moguću praksu za razvoj softvera (Royce, 1970 navedeno u Bassil, 2012). Slično kao u Awadovoj spomenutoj definiciji u prethodnom poglavlju, Bassil iznosi kako je neophodno da jedna faza završi prije nego što sljedeća faza započne. Međutim, Basil navodi kako se svaka faza može ponavljati neograničen broj puta, sve dok nije usavršena.

To donekle pripisuje svojstvo iterativnosti, barem unutar pojedinih faza, što nije spomenuto u Awadovoj definiciji koji iterativnost u pojedinim fazama spominje tek u modelu unificiranog razvoja. Slika 5 predstavlja kaskadni slijed faza karakterističan za vodopadni model koji, ako ga usporedimo sa slikom 1 koja prikazuje isporuku projekta prema tradicionalnoj metodologiji, prati i zadovoljava sve propise tradicionalne metodologije.

Slika 5 Vodopadni model



Izvor: Ilustracija autora

U ovom slučaju, posljednja faza je nazvana održavanje, jer u terminima softvera, rijetko je slučaj da se po završetku projekta softver više neće naknadno održavati.

3.3. Glavne značajke vodopadnog modela

Prateći faze prikazane na slici 5, mogu se i opisati ključne značajke karakteristične za svaku pojedinu fazu ciklusa.

3.3.1. Analiza zahtjeva

Analiza zahtjeva poznata je i kao specifikacija softverskih zahtjeva, analiza zahtjeva mora završiti kompletnim i sveobuhvatnim opisom ponašanja softvera kojeg treba razviti. U proces se trebaju uključiti i sistemski i poslovni analitičari kako bi definirali funkcionalne i nefunkcionalne zahtjeve. Obično se funkcionalni zahtjevi definiraju kroz slučajeve uporabe koji opisuju interakciju korisnika sa softverom. Uključuju zahtjeve kao što su svrha softvera, obuhvat, perspektiva, funkcionalnosti, atributi, karakteristike korisnika koji će ga koristiti, specifikacije funkcionalnosti te zahtjevi sučelja i baze podataka. S druge strane, nefunkcionalni zahtjevi se odnose na različite kriterije, ograničenja, limitacije koji imaju utjecaja više na dizajn proizvoda, nego na njegovo ponašanje. Uključuje svojstva kao što su pouzdanost, skalabilnost, provjerljivost, dostupnost, održivost, performanse i standarde kvalitete. Ova faza odgovara fazi inicijacije prema tradicionalnoj metodologiji (Basil, 2011). Kannan et al. (2014) također navode kako je ova faza najpodložnija greškama, vremenski dugotrajna, ali i troškovno najskuplja faza. Faza analize zahtjeva treba ultimativno završiti funkcijskom specifikacijom koja sadrži odgovore na pitanja što sustav treba činiti te služi kao „ugovor“ između ljudi koji će sustav implementirati i njegovih krajnjih korisnika/kupaca.

3.3.2. Dizajn

Dizajn je proces planiranja i rješavanja problema za softversko rješenje. Podrazumijeva da se programeri i dizajneri usuglase oko plana rješenja koje uključuje dizajn algoritma, dizajn arhitekture softvera, konceptualnu shemu baze podataka, dizajn logičkih dijagrama, dizajn koncepta, grafički dizajn korisničkog sučelja te definiciju strukture podataka (Basil, 2011).

Ova faza odgovara fazi planiranja prema tradicionalnoj metodologiji. Dok se u fazi analize zahtjeva definira odgovor „što“ sustav treba raditi, s fazom dizajna, a kasnije i implementacije se kreće definirati „kako“ će taj sustav biti implementiran i na kraju slučaja, „kako“ će raditi.

3.3.3. Implementacija

Odnosi se na realizaciju poslovnih zahtjeva i funkcijske specifikacije u konkretni program, bazu podataka, web stranicu ili drugu programsku komponentu kroz programiranje i razvoj. Ovo je faza u kojoj se producira stvarni kod te se spaja u operativnu aplikaciju te se kreiraju baza podataka i tekstualne datoteke. Drugim riječima, svi zahtjevi i nacrti se pretvaraju u produkcijsko rješenje (Basil, 2011). Kannan et al. (2014) navode još jednu karakteristiku faze implementacije, a to su jedinični testovi[19]. Kod jediničnog testiranja, svaki modul se testira neovisno o cjelini. Obično se u tom procesu testiranja sustava vrše dorade tj. piše se novi kod kako bi se otklonili pronađeni problemi u radu sustava. Ako se integracija modula kasnije odradi dobro, sustav ne bi trebao imati značajnijih greški tj. problema u radu kada se u potpunosti dovrši.

3.3.4. Testiranje

Jedinični testovi ipak ne pružaju odgovor na pitanje kako će različiti moduli softvera funkcionirati u cjelini, odnosno kakva će biti interakcija među njima. Sistemsko testiranje se bavi upravo pitanjem kako će softver funkcionirati kao cjelina. Efektivno testiranje će osigurati veću kvalitetu softvera, zadovoljnije kupce, manje troškove održavanja te točnije i pouzdanije rezultate (Kannan et al., 2014). Faza testiranja unutar vodopadnog modela poistovjećuje se s fazom kontrole propisanom u tradicionalnoj metodologiji.

3.3.5. Održavanje

Faza održavanja kreće nakon što se softver pusti u produkciju tj. postane dostupan krajnjim korisnicima na korištenje. Stellman i Greene (2005) iznose kako se softver nakon razvoja, u fazi održavanja, dodatno rafinira, ispravljaju se novonastale greške te se popravljaju njegove performanse i kvaliteta. Osim toga, iznose i ostale aktivnosti održavanja kao što su prilagodba

[19] eng. *unit tests*

softvera okruženju ili prilagodba novim korisničkim zahtjevima kao i povećavanje pouzdanosti softvera, koje se mogu, po potrebi, izvršavati u ovoj fazi.

Iako je održavanje posljednja faza u kojoj su aktivnosti projektnih članova često svedene na minimum, svakako ju se ne smije previdjeti i zanemariti. Ova će faza, naime, trajati sve dok se proizvedeni softver ne povuče s tržišta ili dok ne dođe do potrebe za značajnim promjenama sustava koje će iziskivati pokretanje sasvim novog projekta.

3.4. Prednosti i nedostaci vođenja projekta prema vodopadnom modelu

Prednosti koje izvođenje projekata prema vodopadnom modelu donose za uspješnost projekta su zasigurno velike, zbog čega je taj model i dobio zasluženu popularnost. No, kako bi se uspješno mogla provesti analiza modela, svakako je potrebno pogledati i potencijalne prijetnje tj. slabosti koje mogu proizaći primjenom ovog modela.

3.4.1. Prednosti primjene vodopadnog modela u vođenju softverskih projekta

Zasigurno je ključna prednost bilo kojeg modela, pa time i vodopadnog, ta što modeli pružaju strukturu za organizaciju i kontrolu softverskog projekta te su svakako korak prema naprijed od bilo kakvih nestrukturiranih procesa. Osim toga, Kannan et. al. (2014) iznose sljedeće ključne prednosti vodopadnog modela:

1. Detalji u dizajnu i greške se mogu primijetiti prije nego što se softver krene programirati, omogućavajući time znatne uštede na vremenu u kasnijim fazama životnog ciklusa projekta.
2. Vodopadni model iziskuje pripremu odgovarajućeg tehničkog dokumenta (funkcijska specifikacija) koja omogućava kupcu/naručitelju lakše razumijevanje onoga za što je softver namijenjen.
3. Ako se procedure prate adekvatno, troškovi i vremenske procjene se mogu precizno procijeniti.

4. Iznimna pozornost se pridaje izradi dokumentacije. Pomoću nje se svaki novi projektni član može lako pridružiti projektnom timu i nadopuniti znanje o cilju i napretku projekta.
5. Vodopadni model je najbolji model za projekte manjeg obuhvata jer zahtjeva manje resursa u usporedbi s ostalima.
6. Omogućava departmentalizaciju i olakšava menadžersku kontrolu. Postavljanjem rasporeda za svaku pojedinu fazu, moguće je dovršiti proizvod unutar zadanog vremena.
7. Projektni plan se može iskoristiti za iste ili slične projekte u budućnosti

Osim razvoja softvera, Kannan et. al. (2014) iznose da je primjena vodopadnog modela najprikladnija za projekte kojima je krajnji proizvod usluga ili neki drugi nematerijalni proizvod poput dizajna i kreativnog pisanja.

3.4.2. Nedostaci primjene vodopadnog modela u vođenju softverskih projekta

Iz karakteristika vodopadnog modela proizlaze i njegove slabosti, odnosno ograničenja zbog kojih se je nekada teško pridržavati „pravila“ propisanih modelom. Kennan et. al. (2014) iznose sljedeće nedostatke:

1. Vodopadni model zahtjeva da se svi zahtjevi specificiraju na samom začetku projekta što je u praksi gotovo pa nemoguće jer je priroda kupaca takva da često mijenjaju zahtjeve. Osim toga, ako programeri i osobe koje su zadužene za izradu specifikacije nisu iznimno vješti u svojim područjima, teško je unaprijed procijeniti sve korake koje je potrebno provesti u pojedinoj fazi prije nego što ona stvarno započne.
2. Pošto je model linearan, ne dozvoljava značajnije promjene.
3. Prvi djelotvoran proizvod postaje dostupan tek u kasnijim fazama životnog ciklusa projekta.
4. Vodopadni model se ne može skalirati na veće projekte jer su oni rijetko kada sekvencijalni.
5. Ne uključuje rizike i nesigurnost. S obzirom na to da ne postoje iteracije i povratne informacije, ne postoji ni način da se inicijalni nedostaci isprave u kasnijim fazama
6. Teško je mjeriti napredak u svakoj projektnoj fazi.

7. Integracija se odvija na samom kraju što ne pomaže u identificiranju izazova i „uskih grla“ u ranijim fazama projektnog ciklusa.
8. Puno se vremena troši čekajući da pojedina faza završi. Primjerice, dok dizajneri dizajniraju proizvod, programeri nemaju što raditi.
9. Za projekte većih obuhvata, izrada funkcijske specifikacije može potrajati iznimno dugo
10. Koordinacija posla za velike projektne timove koji grade različite komponente softvera iziskuje visoki level menadžiranja, koje ovako jednostavan, sekvencijalni model često ne može garantirati.

Uzevši u obzir navedene činjenice, primjena vodopadnog modela u vođenju projekta razvoja softverskog proizvoda ima smisla u slučajevima kada (Kennan et. al. 2014):

- Je dostupna jasna slika krajnjeg proizvoda.
- Su zahtjevi jasno definirani i razumljivi te nisu podložni promjenama.
- Je u glavnom interesu finalni proizvod, a ne vrijeme potrebno za njegovu izradu.

Velika je opasnost u odstupanju od projektnog rasporeda, ako pojedina faza potraje duže od predviđene, resursi predviđeni za sljedeću fazu (pr. radnici) mogu biti neko vrijeme neiskorišteni. S druge strane, ako faza završi, a da se nisu ispunili propisani zahtjevi, proizvod se može nastaviti razvijati u nezadovoljavajućem obliku. Tek kada su naručitelji projekta sigurni da mogu izbjeći takve scenarije je odabir vodopadnog modela ispravan.

4. SCRUM UPRAVLJAČKI OKVIR KAO NAJPOPULARNIJI AGILNI NAČIN VOĐENJA RAZVOJA SOFTVERSKIH PROIZVODA

Kao što je izneseno u prethodnom poglavlju, tradicionalna metodologija i njen najpopularniji vodopadni model su se pokazali idealnima za projekte manjeg obuhvata i kompleksnosti kod kojih je relativno lako unaprijed definirati sve karakteristike koje bi finalni proizvod trebao imati. Međutim, s obzirom na dinamiku današnjeg poslovnog okruženja, poduzeća konstantno mijenjaju svoje softverske zahtjeve kako bi se prilagodile novom okruženju. Osim brze prilagodljivosti softvera, očekuje se i njegova brza isporuka jer je konkurencija ponuđača softverskih riješenja sve veća i veća. U tom aspektu, tradicionalni, planski orijentirani modeli ne uspijevaju više zadovoljiti kriterije, a tradicionalnu metodologiju je naslijedila agilna metodologija čiji modeli rješavaju upravo probleme zadovoljavanja korisnikovih zahtjeva kroz ranu i kontinuiranu isporuku softvera.

4.1. Glavne značajke agilnog razvoja softverskih proizvoda

Prema Highsmithu i Cockburnu (2001), ono što je novo u vezi agilnih metoda nisu prakse koje koriste, već to što prepoznaju ljude kao primarne pokretače uspjeha projekta, zajedno s intenzivnom usredotočenošću na učinkovitost i upravljivost. Nadalje, Awad (2005) tvrdi kako iz te činjenice proizlaze nove vrijednosti i principi koji definiraju agilni pogled na svijet. Highsmith i Cockburn nadalje definiraju agilnost (Goldman et. al., 1995, navedeno u Highsmith i Cockburn, 2001) kao sveobuhvatan odgovor na poslovne izazove kojima je cilj profitirati od brzo mijenjajućih, kontinuirano fragmentiranih, globalnih tržišta za proizvodima i uslugama visoke kvalitete, visokih performansi koje su istovremeno prilagođene kupcima.“ Termin „agilno“ populariziran je u djelu „Proglas o metodi agilnog razvoja softvera“. U njemu su izneseni temeljni principi agilnog razvoja kao i njegovih dvanaest načela. Beck et. al. (2001) u njemu navode: „Tražimo bolje načine razvoja softvera razvijajući softver i pomažući drugima u njegovom razvoju. Takvim radom smo naučili da više cijenimo:

- Ljude i njihove međusobne odnose, nego procese i oruđa.
- Upotrebljiv softver, nego iscrpnu dokumentaciju.

- Suradnju s naručiteljem, nego pregovaranje oko ugovora.
- Reagiranje na promjenu, nego ustrajanje na planu.

Nadalje, Proglasom se definira i 12 ključnih načela na kojima se proglas zasniva, a to su:

1. Najvažnije nam je zadovoljstvo naručitelja koje postižemo ranom i neprekinutom isporukom softvera koji nosi vrijednost.
2. Spremno prihvaćamo promjene zahtjeva, čak i u kasnijoj fazi razvoja. Agilni procesi upređu promjene da naručitelju stvore kompetitivnu prednost.
3. Često isporučujemo upotrebljiv softver, u razmacima od nekoliko tjedana do nekoliko mjeseci, nastojeći da razmak bude čim kraći.
4. Poslovni ljudi i razvojni inženjeri moraju svakodnevno zajedno raditi tijekom cjelokupnog trajanja projekta.
5. Projekte ostvarujemo oslanjajući se na motivirane pojedince. Pružamo im okruženje i podršku koja im je potrebna i prepuštamo im posao s povjerenjem.
6. Razgovor uživo je najučinkovitiji način prijenosa informacija razvojnom timu i unutar tima.
7. Upotrebljiv softver je osnovno mjerilo napretka.
8. Agilni procesi potiču i podržavaju održivi razvoj. Pokrovitelji, razvojni inženjeri i korisnici trebali bi moći neograničeno dugo zadržati jednak tempo rada.
9. Neprekinuti naglasak na tehničkoj izvrsnosti i dobar dizajn pospješuju agilnost.
10. Jednostavnost – vještina povećanja količine posla kojeg ne treba raditi – je od suštinske važnosti.
11. Najbolje arhitekture, projektne zahtjeve i dizajn, stvaraju samo–organizirajući timovi.
12. Tim u redovitim razmacima razmatra načine da postane učinkovitiji, zatim usklađuje i prilagođava svoje ponašanje.

Većina agilnih tehnika se već koristila prije nastanka Proglasa, ali tek su se s njim one grupirale u izvediv radni okvir. Kako Highsmith i Cockburn navode (2001), agilne metodologije naglašavaju dva koncepta: neumoljivu iskrenost radnog softvera i učinkovitost ljudi koji rade zajedno s dobrom voljom. Radni softver govori programerima i sponzorima što stvarno imaju pred sobom, za razliku od obećanja što će imati ispred sebe u budućnosti. Takav radni softver može se može isporučiti, mijenjati ili brisati, ali je uvijek stvaran. S druge strane, učinkovito korištenje ljudi povećava upravljivost, brzinu i uštedu troškova. Ljudi mogu prenositi ideje brže ako to rade licem u lice nego pisanjem i čitanjem dokumentacije. Nekoliko dizajnera koji sjede zajedno mogu proizvesti bolji dizajn nego što bi svaki mogao

producirati pojedinačno. Kada programeri razgovaraju s kupcima i sponzorima, mogu otkloniti poteškoće, prilagoditi prioritete i ispitati alternativne načine na način na koji ne bi bilo moguće da ne surađuju.

4.1.1. Karakteristike agilnog razvoja softverskih proizvoda

Kako bi se agilni razvoj softvera mogao usporediti s tradicionalnim, potrebno je navesti ključne karakteristike na kojima se zasnivaju metodologije, modeli i načini razvoja koje smatramo agilnima.

4.1.1.1. Ljudski činitelj

Pola principa iz Proglasa o metodi agilnog razvoja softvera se odnosi na ljudski faktor (četiri su temeljna principa). Awad (2015) iznosi da je imati vješte i iskusne ljude u timu ključan faktor za agilne metodologije. Uključivanje domenskih stručnjaka u projektne timove daje ostalim članovima tima brzu povratnu informaciju o implikacijama njihovog dizajna na krajnje korisnike. Suradnja s naručiteljem je prema Awadu još jednan izvrstan faktor zbog toga što daje naručitelju priliku da prati napredak razvoja kao i priliku da promjeni smjer razvoja tijekom svake od faza iteracija. Takva razina predanosti kupcu/naručitelju čini agilnu metodologiju privlačnijom od tradicionalne.

4.1.1.2. Adaptivnost

Kada pogledamo četiri temeljna principa, dva se principa, kao što je navedeno, odnose na ljudski faktor, a uz njih se navodi reagiranje na promjenu i upotrebljiv softver umjesto iscrpne dokumentacije. Prema Fawleru (2004, navedeno u Awad, 2005) sudionici u agilnom procesu se ne boje promjene, dapače, promjene su dobrodošle u bilo kojoj fazi projekta. Prema njemu, promjene zahjeva se smatraju dobrima jer znače da je tim naučio više o tome što je potrebno kako bi se zadovoljilo tržište.

Da bi se moglo reagirati na promjene u pravo vrijeme, potrebno je imati funkcionirajući softver u što ranijoj fazi razvoja, kako bi se promjene mogle kontinuirano uvoditi u kod. Zbog toga agilni procesi poprimaju nelinearan oblik te ih se smatra empirijskima.

4.1.1.3. Empirijski procesi

U inženjerstvu, procesi su definirani ili se odvijaju empirijski. Drugim riječima, definirani procesi su oni koji se, jednom kada se pokrenu, mogu voditi i završiti pri tome producirajući svaki put identičan rezultat. Razvoj softvera često ne može biti smatran definiranim procesom jer se puno toga može promijeniti za vrijeme razvoja. Laurie Williams (2003, navedeno u Awad, 2005) smatra da je teško moguće da će neki set predefiniranih koraka voditi k željenom, predviđenom ishodu jer se mijenjaju zahtjevi, tehnologija, ljudi dolaze i odlaze s projekta i tako dalje.

4.1.1.4. Decentralizirani pristup

Integracija decentraliziranog stila vođenja može značajno utjecati na softverski projekt jer može uzrokovati znatne uštede na vremenu od autokratskih menadžerskih procesa. Agilni razvoj razvijateljima dopušta donošenje odluka. To ne znači da tim preuzima ulogu menadžmenta. Menadžment je potreban kako bi uklonio sve zapreke koje stoje na putu napretka. Međutim, menadžment prepoznaje stručnost tehničkog tima i dopušta mu donošenje tehničkih odluka (Awad, 2005).

4.1.1.5. Uključenost kupaca i kolaboracija

Kupci su aktivno uključeni u procese u agilnom razvoju softverskih proizvoda. Uključenost kupaca pomaže u mitigiranju jednog od najkonstantnijih problema prilikom razvoja softvera, a to je problem odstupanja inicijalnih zahtjeva kupaca i njihovih očekivanja od proizvoda kada se on dovrši. Na ovaj način, kupac ima bolju viziju nastajućeg proizvoda. Uz mogućnost vizualizacije funkcionalnosti koja dolazi na temelju uvida u ono što je do određenog trenutka izgrađeno, kupci razvijaju i bolje razumijevanje vlastitih potreba i načina kako bi se izrazili pri definiranju zahtjeva. Uključenost kupaca u procese ide u prilog i ljudskom faktoru, koji nije ograničen samo na razvojni tim, već i na uključenost naručitelja i kupaca u svim fazama razvoja. Agilne metodologije podupiru kontinuiranu komunikaciju uživo za sve dionike projekta.

4.1.2. Usporedba karakteristika tradicionalne i agilne metodologije u razvoju softverskih proizvoda

S obzirom na karakteristike metodologija koje su do sada navedene, u tablici 4 su ukratko prikazani ključni parametri pojedinih metodologija i njihove karakteristike s obzirom na pojedinu metodologiju.

Tablica 3 Usporedba primjene tradicionalne i agilne metodologije u vođenju razvoja softverskih proizvoda

Parametar	Tradicionalna metodologija	Agilna metodologija
Pristup	Prediktivan	Adaptibilan
Mjere uspjeha	Pridržavanje plana	Poslovna vrijednost
Veličina projekta	Veliki projekti	Manji projekti
Stil vođenja	Autokratski	Decentralizirano
Perspektiva promjena	Promjena održivosti	Promjena adaptibilnosti
Kultura	Naredbe i kontrola	Vodstvo i kolaboracija
Dokumentacija	Visoka	Niska
Orijentiranost	Procesi	Ljudi
Ciklusi	Ograničeni	Brojni
Domena	Predvidiva	Nepredvidiva/Eksploratorna
Planiranje	Opsežno	Minimalno
Povrat na investiciju	Na kraju projekta	Rano u projektu
Veličina tima	Velika	Mala/Kreativna

Izvor: Awad, 2005

S obzirom na navedene parametre, naručiteljima projekta i projektnim sponzorima ne bi trebao biti problem odabrati pojedinu metodologiju s obzirom na projektne karakteristike i zahtjeve. Međutim, kontekst projekta će svakako odlučiti o tome koja je metodologija najprikladnija potrebama projektu. Primjerice, veliki projekti ne moraju nužno biti vođeni po tradicionalnoj metodologiji, ali će primjena agilne unijeti dodatnu razinu kompleksnosti u sve projektne faze i za sve članove tima. Ako projektni članovi nisu spremni na kontinuiranu i aktivnu kolaboraciju, agilna metodologija svakako neće biti logičan izbor, bez obzira na veličinu tima. S toga, navedene parametre treba uzeti u obzir prilikom donošenja odluke, ali ih se ne treba shvatiti kao eliminirajuće faktore.

4.1.3. Metodologije i upravljački okviri agilnog razvoja softverskih proizvoda

Isto kao kod tradicionalne metodologije i unutar agilne je došlo do podjela na više metodologija tj. razvojnih okvira upravo zbog dinamičnog poslovnog okruženja i različitosti zahtjeva kojima projektni timovi moraju udovoljiti. Najpoznatiji oblici su:

- Ekstremno programiranje
- Scrum
- Razvoj vođen funkcionalnostima[20]

4.1.3.1. Ekstremno programiranje

Ekstremno programiranje je proces kojeg mogu okarakterizirati kratki razvojni ciklusi, inkrementalno planiranje, kontinuirane povratne informacije, oslanjanje na komunikaciju i evolucijski dizajn (Beck, 2004 navedeno u Awad, 2005). Ekstremno programiranje se može smatrati metodologijom. Lindstrom i Jeffries (2006) iznose kako ekstremno programiranje dijeli vrijednosti koje zagovara Proglasu o metodi agilnog razvoja, ali ide i dalje od toga definirajući pri tome određeni skup praksi. Dok se većina popularnih metodologija bavi pitanjima koje sve prakse trebaju na softverskim projektima, ekstremno programiranje nastoji odgovoriti na pitanje koji je najjednostavniji set praksi koje su potrebne u razvoju i što se može napraviti kako bi se ograničile potrebe za tim istim praksama.

Sažetak uvjeta i praksi ekstremnog programiranja prikazan je u nastavku (Awad, 2005):

- Planiranje – Programer procjenjuje rad potreban za implementaciju naručiteljevih zahtjeva te naručitelj odlučuje o obuhvatu i vremenu na temelju te procjene.
- Mali/kratki izlasci – Softverske aplikacije se razvijaju u serijama malih i često ažuriranih verzija. Nove verzije izlaze po dnevnoj, tjednoj ili mjesečnoj bazi.
- Metafore – Sustav je definiran kroz set metafora između naručitelja i programera koje opisuju kako sustav radi.
- Jednostavan dizajn – Naglasak je na dizajniranju najjednostavnijeg mogućeg rješenja. Svaka nepotrebna kompleksnost i višak koda se promptno miče.

[20] eng. *Feature driven development*

- Refaktoring – Uključuje restrukturiranje sustava tako da se izbacuju duplikacije, poboljšava komunikacija, pojednostavljuje i povećava fleksibilnost sustava, ali bez da se mijenjaju njegove funkcionalnosti.
- Programiranje u paru – Produkcijski kod pišu dvije osobe za jednim računalom.
- Kolektivno vlasništvo – Ni jedna osoba ne posjeduje niti je odgovorna za pojedine segmente koda tj. svatko može promijeniti bilo koji dio koda u bilo koje vrijeme.
- Kontinuirana integracija – Novi dijelovi koda su integrirani u trenutni sustav čim budu spremni. Prilikom integriranja, sustav se gradi na novo te moraju proći svi testovi kako bi se promjene prihvatile.
- 40-sati radni tjedan – Nitko ne smije raditi dva tjedna zaredom prekovremene sate.
- Kupac/naručitelj na licu mjesta – Kupac/Naručitelj mora biti uvijek dostupan razvojnom timu.
- Standardi kodiranja – Pravila kodiranja postoje te ih programeri moraju pratiti kako bi se postigla konzistentnost i poboljšala komunikacija u razvojnom timu.

Životni ciklus ekstremnog programiranja također je podijeljen u faze, točnije njih šest: istraživanje, planiranje, iteracije, produkcija, održavanje te kraj projekta (Awad,2005). Aktivnosti karakteristične za pojedinu fazu razvoja su prethodno već opisane kod drugih modela razvoja softverskih proizvoda.

4.1.3.2. Scrum

Scrum je agilni upravljački okvir namijenjen upravljanju složenim radom, s naglaskom na razvoj softvera, iako se koristi i u drugim poljima te se polako počinje istraživati za druge složene poslove, istraživanja i napredne tehnologije (Wykovski i Wykovska, 2018). Verheyen (2013), a i broja druga literatura, Scrum nazivaju razvojnim okvirom[21] jer se ne sastoji od iscrpnih i formalnih „recepata“ o tome kako dizajnirati i planirati rad, postupke i ponašanje svih članova koji su uključeni u razvoj proizvoda u odnosu na vrijeme, a kamo li kako bi takvi dizajnovi i planovi morali biti dokumentirani, odobreni, pohranjeni itd..S druge strane, metodologije se tipično sastoje od strogih i obaveznih nizova procesa i postupaka koji implementiraju unaprijed definirane algoritme. Kao takve, metodologije uglavnom isključuju kreativnost, autonomiju i mišljenje ljudi s komponentama poput faza, zadataka, obaveznih

[21] eng. *framework*

praksi, tehnika i alata. Kao upravljački okvir, Scrum opisuje uloge i pravila na temelju danih agilnih načela koja pomažu i facilitiraju ljude na nepropisujući način (Verheyen, 2013).

Termin Scrum su prvi puta upotrijebili Hirotaka Takeuchi i Ikujiro Nonaka u svom radu 1986. godine „The New New Product Development Game“. Naime, posudili su termin iz ragbija kako bi naglasili važnost timskog rada u kompleksnom razvoju proizvoda (Vergeyen, 2017). Rising i Janoff (2000) dodatno opisuju Scrum u ragbiju kao tim od osam individualaca. Svi u timu rade zajedno te sudjeluju u prenošenju lopte kroz teren te djeluju kao integrirane jedinice u kojima svaki član tima ima točno definiranu ulogu, ali svi imaju jedan zajednički cilj.

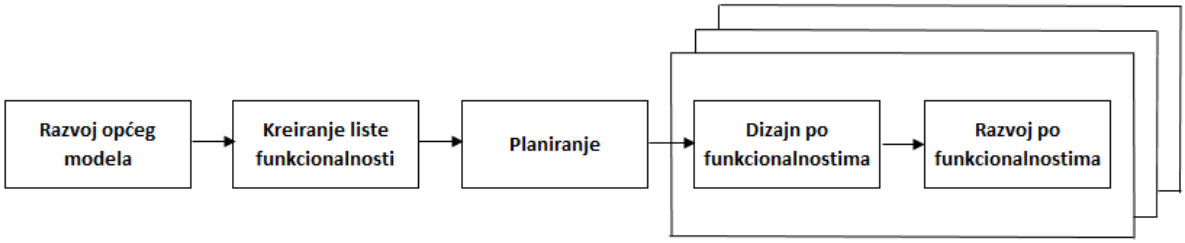
Istraživanje Hirotake Takeuchi i Ikujira Nonake je pokazalo da je moguće postići izvanredan uspjeh kada se timovi ponašaju kao male, samoorganizirajuće cjeline te kada se takvi timovi „hrane“ (izazivaju) ciljevima, a ne zadacima. Timovi mogu dosegnuti vrhunac samo ako im se da prostora da osmisle svoje vlastite taktike za ostvarivanje danih ciljeva (Verheyen, 2017). Stoga, ime Scrum i razlog zašto je dodijeljen jednom razvojnom okviru koji će kasnije postati jedan od najpopularnijih metoda razvoja softverskih proizvoda, proizlazi iz naglaska na timsko djelovanje i ljudskog faktora koji se smatra ključnim za čitavi proces. Scrum, njegove karakteristike i događaji će se opisati opširnije u poglavlju koje slijedi.

4.1.3.3. Razvoj temeljen na funkcionalnostima

Razvoj temeljen na funkcionalnostima[22] je prvi puta bio korišten u projektu razvoja velike i kompleksne bankarske aplikacije u kasnim devedesetima. Za razliku od ostalih metodologija, FDD ne pokriva u cijelosti proces razvoja softvera nego se fokusira na dizajn i fazu razvoja (Palmer i Felsing, 2002 navedeno u Awad, 2005).
Prve tri faze se odvijaju na početku projekta. Zadnje dvije faze su iterativni dio procesa koji podržavaju agilni razvoj s brzim adaptacijama na kasne izmjene zahtjeva i poslovne potrebe. FDD uključuje isporučivanje učestalih i opipljivih verzija tijekom procesa uz precizno monitoriranje napretka i izvještavanje (De Luca, 2005 navedeno u Awad, 2005).

[22] eng. *Feature Driven Development (FDD)*

Slika 6 Razvoj temeljen na funkcionalnostima



Izvor: Palmer i Felsing, 2002, navedeno u Awad, 2005.

Awad na sljedeći način definira faze u procesu razvoja temeljenog na funkcionalnostima:

- Razvoj općeg modela – domenski stručnjak prolazi kroz čitav obuhvat sustava i njegov kontekst te ga prezentira članovima tima i glavnom arhitektu. Razvijaju se dokumentirani zahtjevi kao što su slučajevi uporabe i funkcijske specifikacije.
- Lista funkcionalnosti – predstavlja kategoriziranu listu funkcionalnosti koje podržavaju zahtjeve.
- Kreiranje plana – razvojni tim prioritizira funkcionalnosti i dodjeljuje ih glavnim programerima. Također se postavljaju rasporedi i miljokazi [23] za setove funkcionalnosti.
- Dizajn i razvoj po funkcionalnostima – Dizajn i razvoj su iterativni postupci tijekom kojih timovi produciraju sekvencijalne dijagrame za dodijeljene funkcionalnosti. Takvi dijagrami se predaju programerima koji implementiraju stavke ključne za podršku dizajnu za određene funkcionalnosti. Više različitih timova može istovremeno raditi na različitim funkcionalnostima. Kada se kod dovrši, on se testira te se nakon uspješne iteracije uvrštava u glavnu verziju.

Razvoj temeljen na funkcionalnostima organizira proces razvoja na manje, naručitelju vrijedne, jedinice zvane funkcionalnosti. Zbog toga olakšava praćenje napretka razvoja softvera i izvještavanje o njemu, čineći tako čitav proces transparentnijim. Kupci tj naručitelji mogu tako već iznimno rano u fazi razvoja dobiti opipljive verzije proizvoda s ključnim funkcionalnostima.

[23] eng. *milestones*

4.2. Definicija Scrum upravljačkog okvira

U prethodnom poglavlju se definirao Scrum kao upravljački okvir. U ovom poglavlju je cilj detaljno opisati Scrum upravljački okvir te sve što on propisuje. S obzirom na dinamičnost trenutne okoline u kojoj se razvijaju softverski proizvodi, projektni zahtjevi ostaju djelomično nejasni čak i nakon što projekt započne i krene napredovati. Kao što je već navedeno, Scrum je upravljački okvir koji ljudi mogu koristiti u adresiranju kompleksnih problema s ciljem isporuke proizvoda najveće moguće vrijednosti. Schwaber i Sutherland (2017) iznose kako je Scrum zasnovan na teoriji empirijskog upravljanja procesima, odnosno empirizmu. Empirizam tvrdi da znanje dolazi iz iskustva i donošenja odluka na temelju onoga što je poznato. Scrum koristi iterativni i inkrementalni pristup kako bi se optimizirala predvidljivost i kontrola rizika.

4.3. Ključne sastavnice Scruma

Scrum se sastoji od tri razvojne komponente, uključujući uloge, ceremonije (događaje) i artefakte (Schwaber, 2004 navedeno u Cho, 2008).

4.3.1. Scrum tim

Scrum tim sastoji se od Vlasnika proizvoda[24], Razvojnog tima i Scrum Mastera. Scrum timovi su samoorganizirajući i višefunkcionalni. Samoorganizirajući timovi biraju način kako najbolje napraviti svoj posao, umjesto da im netko drugi izvan tima postavlja pravila. Višefunkcionalni timovi imaju sve potrebne vještine za dovršenje posla i ne ovise o drugima koji nisu članovi tima. Model tima u Scrumu je postavljen tako da iz članova tima izvuče što više fleksibilnosti, kreativnosti i produktivnosti. Sve se više pokazuje da je Scrum tim iznimno učinkovit za sve ranije navedene načine korištenja kao i za bilo koji drugi kompleksni posao (Schwaber i Sutherland, 2017).

[24] eng. *Product Owner*

4.3.1.1. Vlasnik proizvoda

Vlasnik proizvoda je odgovoran za uvećavanje vrijednosti proizvoda koji nastaje kao rezultat rada Razvojnog tima. Način na koji se to radi se može jako razlikovati od organizacije do organizacije, među Scrum timovima i pojedincima. Vlasnik proizvoda je jedina osoba odgovorna za upravljanje Popisom stavki[25] koje proizvod treba imati (Schwaber i Sutherland, 2017).

U Scrum vodiču, koji je postao najpopularniji vodič kroz Scrum okvir, Schwaber i Sutherland (2017) definiraju zadatke Vlasnika proizvoda:

- Jasno definiranje stavki na Popisu.
- Slaganje redoslijeda stavki na Popisu kako bi se ciljevi i misije dostigli na najbolji mogući način.
- Optimiziranje vrijednosti posla kojeg Razvojni tim odrađuje.
- Osiguravanje da je Popis stavki vidljiv, transparentan i jasan svima, te da pokazuje na čemu će sljedećem raditi Scrum tim.
- Osiguravanje da Razvojni tim razumije stavke s popisa do razine koja je potrebna.

Također, bitno je naglasiti da je Vlasnik proizvoda jedna osoba te da se ova uloga ne bi smjela dijeliti među više odgovornih osoba. Nužno je izbjeći i bilo kakvo preklapanje uloga tj. situaciju u kojoj je jednoj osobi dodijeljeno više uloga. Vlasnik projekta predstavlja želje interesnih skupina (Schwaber i Sutherland, 2017) pa je dobra praksa da ta osoba poznaje i usko surađuje s naručiteljima projekta kako bi mogla definirati i prioritizirati zadatke čime će značajno utjecati na tijek razvoja.

4.3.1.2. Razvojni tim

Razvojni tim se sastoji od profesionalaca koji obavljaju posao na isporuci inkrementa „gotovog“ proizvoda koji se može pustiti u rad na kraju svakog Sprinta (Schwaber i Sutherland, 2017). Nadalje, Schwaber i Sutherland definiraju karakteristike Razvojnog tima

[25] eng. Product backlog

- Sami se organiziraju. Nitko (čak ni Scrum Master) ne naređuje Razvojnom timu kako da pretvore Popis stavki za proizvod u inkremente potencijalno isporučivih funkcionalnosti.
- Razvojni timovi su višefunkcionalni, sa svim vještinama koje su timu potrebne za izradu inkrementa proizvoda.
- Scrum ne prepoznaje titule za članove Razvojnog tima, bez obzira na različita područja koja treba uzeti u obzir kao što su testiranje ili poslovna analiza; ovo pravilo nema iznimki.
- Scrum ne razlikuje pod-timove u Razvojnom timu.
- Pojedini članovi Razvojnog tima mogu imati specijalizirane vještine i područja na koja se fokusiraju, ali odgovornost pripada Razvojnom timu kao cjelini.

Ključna osobina razvojnog tima u Scrumu je ta da su višefunkcionalni. Takvi timovi imaju sve potrebne vještine, znanja i ekspertizu unutar tima koji su im potrebni za završavanje definiranog posla. U tom slučaju, razvojni tim za dovršavanje radnih zadataka ne treba nikakvu eksternu pomoć (od osoba izvan tima). Upravo karakteristika višefunkcionalnosti doprinosi tome da se takvi timovi mogu sami organizirati i preuzeti odgovornost za isporučene stavke.

Optimalna veličina razvojnog tima je između tri i devet članova (Schwaber i Sutherland, 2017). Schwalber i Sutherland to argumentiraju na način da manje od tri člana ograničava interakciju i rezultira manjom produktivnošću, dok više od devet članova tima iziskuje previše koordinacije te tada timovi generiraju previše kompleksnosti da bi kontrola procesa bila korisna.

4.3.1.3. Scrum Master

Scrum Master obnaša funkciju koja je relativno uska u obuhvatu, a poprilično široka u utjecaju na čitavo poduzeće. U praksi, Scrum Master radi „iza kulisa“ te nije uključen ni u kakvu ideaciju proizvoda niti proizvodnu strategiju. Scrum Master djeluje više kao kanal između vlasnika proizvoda/linije poslovanja i razvojnih timova tj. kao voditelji projekata. Zbog činjenice da su agilni procesi upotpunosti ovisni o ljudima i njihovoj kolaboraciji,

Scrum master mora posjedovati i njegovati svoje vještine rada s ljudima te poznavati najažurnije alate i metode koje će primjenjivati u radu.
Scrum master pruža usluge Vlasniku proizvoda, Razvojnom timu i samom poduzeću te ih je zbog tog razloga mnogo, a neke od navedenih u Scrum vodiču (Schwaber i Sutherland, 2017) su sljedeće:

- Osigurava da svi članovi Scrum tima razumiju ciljeve, opseg i domenu proizvoda na najbolji mogući način.
- Pronalazi tehnike za učinkovito upravljanje Popisom stavki koje proizvod treba imati.
- Pomaže Scrum timu kod razumijevanja potrebe za jasnim i konciznim stavkama s Popisa.
- Pomaže kod razumijevanja planiranja proizvoda u iskustvenom okruženju.
- Trener je Razvojnom timu prema samoorganizaciji i višefunkcionalnosti.
- Uklanja prepreke koje Razvojnom timu stoje na putu.
- Vodič i trener je poduzeću u prihvaćanju Scruma.
- Planira uvođenje Scruma u okviru poduzeća.

S obzirom da je tim u Scrumu samoorganizirajući, Scrum master nije zadužen za organizaciju procesa i događaja, već samo za njihovo facilitiranje te uklanjanje prepreka koje koče tim u napretku. Razumijevanje toga je ključno za razlikovanje uloga projektnog menadžera i Scrum mastera.

4.3.2. Scrum ceremonije (događaji)

S obzirom da je Scrum upravljački okvir, propisuje određena događanja koja moraju poprimiti određeni oblik, redovitost i trajanje kako bi bila u skladu s onime što Scrum propisuje. Kao što je navedeno u ulogama Scrum mastera, iznimno je važno da je osoba koja obnaša tu funkciju adekvatno obrazovana te da može služiti timu kao vodič i trener kod uspostavljanja Scruma u poduzeću. Promjene s tradicionalnijih na agilne razvoje su svakako značajne i iziskuju prilagodbe svih osoba uključenih u procese te je takvu migraciju svakako nemoguće odraditi „preko noći“. Jedan od najpopularnijih događaja Scruma, koji obično biva i prvim implementiranim u procese je Sprint.

4.3.2.1. Sprint

Srce Scruma je Sprint, vremensko ograničenje od jednog mjeseca ili kraće tijekom kojeg se kreira "Gotov" i upotrebljiv proizvod, koji se potencijalno može pustiti u rad. Najbolje je da Sprintevi imaju uvijek isto trajanje tijekom razvoja proizvoda. Novi Sprint započinje odmah nakon završetka prethodnog (Schwaber i Sutherland, 2017). Schwaber i Sutherland također definiraju radnje koje se smiju/ne smiju raditi tijekom trajanja Sprinta:

- Ne rade se nikakve promjene koje bi mogle ugroziti Cilj sprinta.
- Željena kvaliteta se ne umanjuje.
- Vlasnik proizvoda i Razvojni tim mogu razjasniti i ponovno dogovoriti opseg posla, kako se saznaju nove informacije.

„Idealno“ trajanje sprinta bi bilo u intervalu od dva tjedna, ali izabrani vremenski interval će svakako ovisiti o prirodi proizvoda koji se isporučuje te učestalosti potrebe da kupac/naručitelj sudjeluje u njegovom pregledu. Dakle, svaki Sprint se može smatrati posebnim projektom, te može uključivati sve faze životnog ciklusa u razvoju proizvoda, od dizajna do isporučivanja inkrementalnog proizvoda, dakako s manjim obuhvatom od obuhvata čitavog projekta.

Postoji nekoliko ključnih događaja koji se odvijaju tijekom sprinta, a to su: Planiranje sprinta[26], Dnevni sastanci[27], Pregled Sprinta[28] i Osvrt na sprint[29] (Schwaber i Sutherland, 2017).

4.3.2.2. Planiranje Sprinta

Planiranje sprinta je događaj u kojem se planira posao kojeg treba obaviti u nadolazećem sprintu. Na njemu sudjeluje Razvojni tim, Scrum master i Vlasnik proizvoda te se održava početkom svakog Sprinta tj. iteracije (Cervone, 2010).
Duljina trajanja Planiranja sprinta ovisi o određenom vremenskom intervalu trajanja čitavog Sprinta. Schwaber i Sutherland (2017) definiraju da planiranje sprinta ne traje duže od osam

[26] eng. *Sprint planning*),
[27] eng. *Daily Scrum*
[28] eng. *Sprint review*
[29] eng. *Sprint Retrospective*

sati za jednomjesečni Sprint. S obzirom na taj vremenski okvir, Sprintevi koji traju primjerice dva tjedna bi u praksi trebali imati četverosatno Planiranje sprinta. Nadalje, Schwaber i Sutherland iznose temeljna pitanja na koja Planiranje sprinta odgovara:

- Kakav inkrement može biti isporučen kao rezultat predstojećeg Sprinta?
- Kako će se to postići?

Na prvo pitanje se odgovara u prvom dijelu Planiranja sprinta u kojem tim definira Popis stavki za proizvod koji zapravo predstavlja popis projektnih zahtjeva. Nakon toga tim zajedno determinira cilj Sprinta koji će predstavljati formalni ishod tog određenog Sprinta. U drugom dijelu sastanka, fokus je na kreiranju Popisa stavki za Sprint (Cervone, 2010). Schwaber i Sutherland (2017) iznose kako u tom, drugom dijelu sastanka Razvojni tim odlučuje kako će tijekom Sprinta izgraditi odabranu funkcionalnost u "Gotovi" inkrement proizvoda. Stavke s Popisa stavki koje proizvod treba imati i plan za njihovu isporuku se zajedno zovu Popis stavki za sprint.

Do kraja sastanka Razvojni tim dekomponira posao koji se planira za prvih nekoliko dana Sprinta, često usitnjeno do razine zadataka kojima treba jedan dan da se dovrše, nekad i kraće. Razvojni tim se sam organizira kako će preuzeti i napraviti posao s Popisa stavki za sprint, i tijekom Planiranja sprinta i kako je potrebno tijekom samog Sprinta. Do kraja Planiranja sprinta Razvojni tim treba biti u mogućnosti objasniti Vlasniku proizvoda i Scrum Masteru kako namjeravaju raditi kao samoorganizirajući tim da bi postigli Cilj sprinta i kreirali očekivani Inkrement (Schwaber i Sutherland, 2017).

Ključan faktor za uspjeh čitavog Sprinta i ostvarenje njegovog cilja je uspješno provedeno Planiranje sprinta. Planiranje će biti toliko uspješno koliko je Razvojni tim sposoban iskomunicirati ključne stavke te podijeliti rad među članovima ovisno o njihovoj ekspertizi. Svakako treba imati na umu da će timovi koji po prvi puta kreću raditi zajedno trebati više vremena te više pomoći Scrum mastera i Vlasnika proizvoda od uhodanih timova koji već dobro poznaju ekspertizu i brzinu svakog člana razvojnog tima. Procjene zadataka i funkcionalnosti koje se uvode u Sprint će također bivati sve točnijima što Razvojni tim ima više iskustva u zajedničkom radu.

4.3.2.3. Dnevni sastanci

U mnogim projektima koji se vode po Scrumu, ali ne nužno i svima, radni dan započinje Dnevnim sastankom. Ovaj sastanak, koji obično traje najviše petnaest minuta, održava se svaki dan između Scrum mastera (koji facilitira sastankom) i Scrum tima (Cervone, 2010).

Samu strukturu sastanka definira Razvojni tim. Može se održavati na različite načine sve dok pomaže zadržati fokus na napredak prema Cilju sprinta. Neki Razvojni timovi tijekom sastanka koriste pitanja, dok se drugi više oslanjaju na razgovor (Schwaber i Sutherland, 2017). Schwaber i Sutherland iznose primjere pitanja koji se mogu koristiti:

- Što sam napravio jučer, a da je pomoglo Razvojnom timu dostići Cilj sprinta?
- Što ću napraviti danas da pomognem razvojnom timu dostići cilj sprinta?
- Vidim li kakve prepreke koje mogu onemogućiti mene i razvojni tim u dostizanju Cilja sprinta?

Cervone (2010) navodi da, iako to možda nije odmah jasno, dnevni sastanci nisu sastanci u kojima je cilj rješavati probleme i prikupljati informacije o tome tko kasni sa zadacima, već je njihov cilj praćenje napretka tima te omogućavanje članovima tima da se obvežu jedni drugima kako bi se rad nastavio s najmanje potencijalnih prepreka.

Cilj dnevnih sastanaka je svakako poboljšati komunikaciju između članova. Osim toga, tim kratkim sastancima se može eliminirati potreba za drugim, dužim i neproduktivnijim sastancima jer svi članovi uhodano prate kako svoj, tako i napredak ostalih članova u timu čime se mogu riješiti neke ključne prepreke u međuovisnostima komponenti koje bi inače zahtijevale puno više vremena za identifikaciju. Scrum master mora svakako paziti da se tim na dnevnim sastancima drži tema iz postavljenih pitanja, te ne zalaze u druge probleme koje ne bi smjeli biti tema dnevnog sastanka.

4.3.2.4. Pregled Sprinta

Pregled Sprinta se odvija na kraju svakog Sprinta. Tijekom sastanka se funkcionalnost koja je implementirana tijekom Sprinta prezentira Vlasniku proizvoda. Ono što najviše razlikuje ovaj

sastanak od sastanaka u tradicionalno vođenim projektima je to da taj sastanak treba biti neformalan te ne smije biti distrakcija članovima tima (Cervone, 2010).

Tijekom Pregleda sprinta Scrum tim i ključne zainteresirane strane zajedno prolaze kroz Inkrement proizvoda koji je nastao tokom Sprinta. Temeljeno na tim informacijama i svim promjenama na Popisu stavki koje su nastale tokom Sprinta, sudionici zajedno dogovaraju koje sljedeće stvari se mogu napraviti kako bi se povećala vrijednost. Ovo nije statusni već neformalni sastanak, a svrha prezentacije Inkrementa je dobiti povratnu informaciju i potaknuti suradnju (Schwaber i Sutherland, 2017). Schwaber i Sutherland uključuju sljedeće elemente u pregled Sprinta:

- Sudionici su Scrum tim i ključne zainteresirane strane koje poziva Vlasnik proizvoda.
- Vlasnik proizvoda objašnjava koje stavke s Popisa za proizvod su "Gotove", a koje stavke još nisu završene.
- Razvojni tim diskutira što je prošlo dobro tokom Sprinta, na koje su probleme naletjeli i kako su ti problemi bili riješeni.
- Razvojni tim pokazuje posao koji je "Gotov" i odgovara na pitanja o Inkrementu.
- Vlasnik proizvoda diskutira Popis stavki za proizvod u njegovom trenutnom stanju. On ili ona projicira moguće datume dovršetka, temeljeno na dosadašnjem napretku (ako je potrebno).
- Cijela skupina zajedno diskutira što će se sljedeće raditi, tako da Pregled sprinta daje korisne informacije za sljedeće Planiranje sprinta.
- Pregled tržišta ili mogućeg načina korištenja proizvoda može promijeniti koja je sljedeća najvrjednija stvar za napraviti.
- Pregled vremenskog plana, budžeta, potencijalnih mogućnosti i tržišta za sljedeće očekivano izdanje proizvoda.

Pregled Sprinta je događaj na kojem će tim dobiti najviše povratnih informacija o napretku svog posla. Činjenica da jednom mjesečno, a u nekim slučajevima i svakih dva tjedna, članovi Razvojnog tima moraju prezentirati implementirane stavke ostalim zainteresiranim stranama, daje članovima određen stupanj „vlasništva nad proizvodom“. Članovi Razvojnog tima u Scrumu nisu osobe koje rade posao „iza kulisa“ i nikada se niti ne pojave pred naručiteljima/kupcima, već oni kontinuirano zastupaju i prezentiraju svoj rad pred njima.

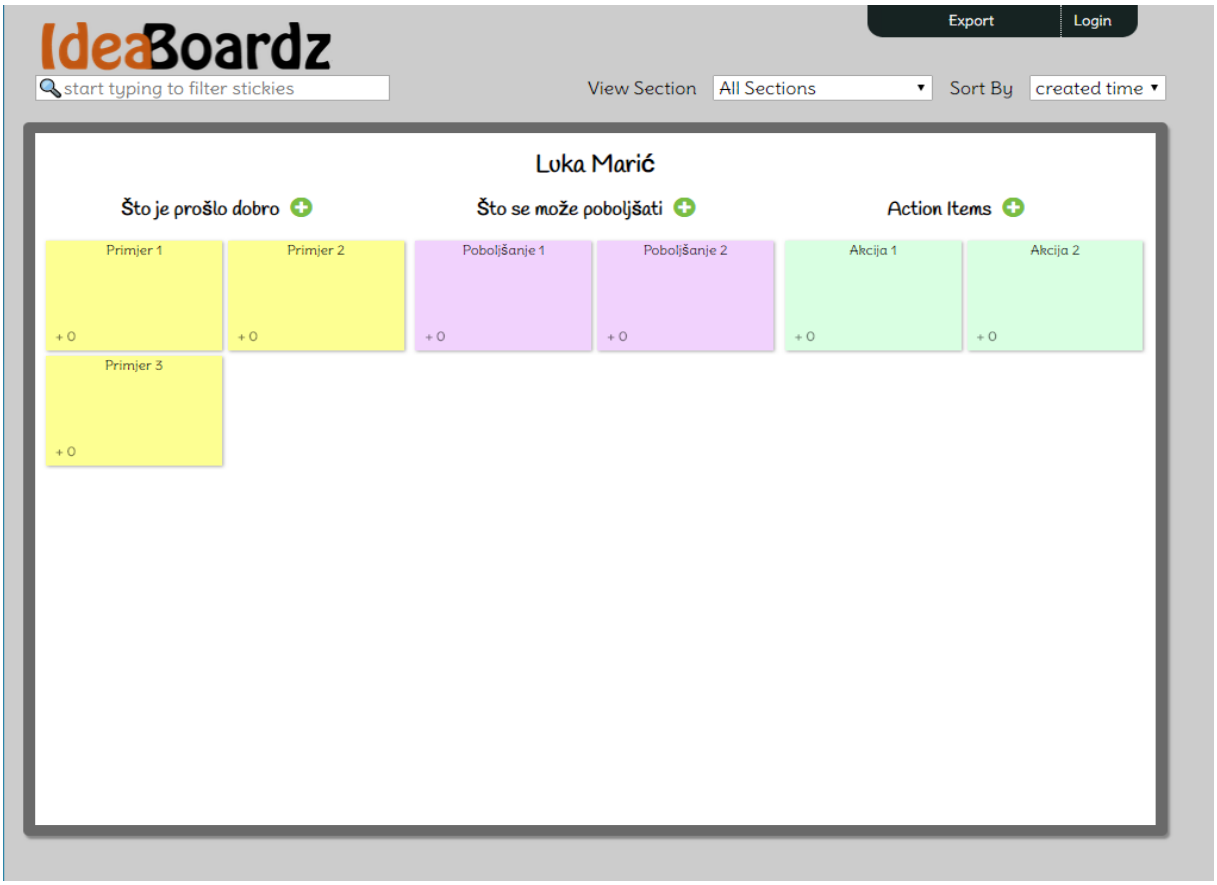
Takav oblik transparentnosti svakako dovodi do ultimativno boljeg proizvoda koji na kraju krajeva više odgovara potrebama tržišta tj. kupaca.

4.3.2.5. Osvrt na Sprint

Osvrt na Sprint je prilika da se Scrum tim osvrne na sebe i svoj način rada i da napravi plan za poboljšanja koja će se primijeniti tokom sljedećeg Sprinta. Osvrt na Sprint se događa nakon Pregleda sprinta, a prije sljedećeg Planiranja sprinta. Njegova svrha je kontrola kako je prošao prethodni Sprint s obzirom na ljude, njihove odnose, procese i alate, prepoznavanje i slaganje po važnosti glavnih stavki koje su prošle dobro i koja su potencijalna unapređenja te izrada plana za uvođenje poboljšanja u način na koji Scrum tim radi svoj posao (Schwaber i Sutherland, 2017).

U Osvrtu na Sprint bi trebali sudjelovati svi koji su uključeni u Sprint, a sigurno je da bi najviše komentara trebali imati Vlasnik proizvoda, Razvojni tim i Scrum master. Jedan od načina prikupljanja informacija u Osvrtu je taj da se povratne informacije podijele na ono što je prošlo dobro, ono što se može poboljšati te akcije koje je potrebno poduzeti kako bi se napredak poboljšao. Postoji mnogo internetskih alata koji olakšavaju prikupljanje ideja članova tima, a jedan od njih je IdeaBoardz prikazan na slici 7.
Alat kao što je IdeaBoardz olakšava anonimno prikupljanje informacija jer se „ploča“ može podijeliti sa svim članovima tima te oni mogu anonimno dopisati svoje komentare. Time se eliminira ključni problem, ako on postoji, koji leži u tome da je nekim članovima neugodno izreći svoje stajalište pred ostalim zainteresiranim stranama. Postoji još velik broj alata i predložaka koji olakšavaju strukturiranje pojedinih Scrum (zadatak Scrum mastera). Iako će većina timova poći za vlastitim riješenjima, najvažnije je da, koji god alat se koristi, bude shvatljiv i prihvatljiv čitavom timu.

Slika 7 Primjer alata za prikupljanje povratnih informacija na Osvrtu na Sprint – IdeaBoardz



Izvor: primjer autora

4.3.3. Scrum artefakti

Zadnja velika komponenta Scruma su artefakti, kojima nazivamo Popis stavki za proizvod, Popis stavki za Sprint i Inkrement. Artefakti u Scrumu predstavljaju posao ili vrijednosti koje daju transparentnost i prilike za kontrolu i prilagodbu. Artefakti definirani u Scrumu su posebno dizajnirani da maksimalno povećavaju transparentnost ključnih informacija tako da svi na jednak način razumiju dati artefakt (Schwaber i Sutherland, 2017).

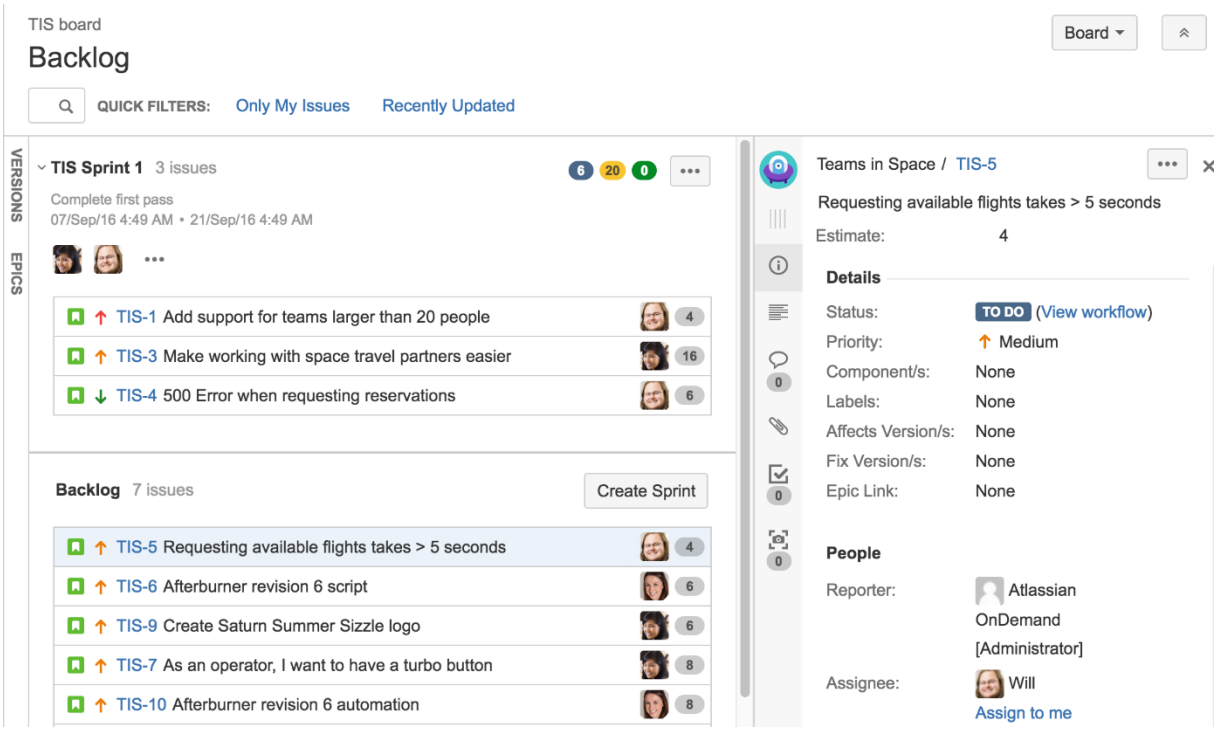
4.1.3.4. Popis stavki za proizvod

Cervone (2010) definira Popis stavki za proizvod kao projektne zahtjeve iskazane putem prioritizirane liste zadataka. Cervone tvrdi da za razliku od tradicionalnih metoda razvoja, vlasništvo nad ovom listom ima Vlasnik proizvoda te on njom upravlja. Takva lista se može

napraviti pomoću bilo kojeg alata za projektni menadžment (primjerice MS-Project), ali može jednako efikasno biti prikazana proračunskom tablicom[30].

Na slici 8 je prikazan izgled jednog takvog popisa u jednom od najpopularnijih alata za vođenje projekata „Jira“.

Slika 8 Primjer Popia stavki za proizvod u alatu „Jira“



Izvor: Confluence, 2019

Na slici je vidljiv popis zadataka (TIS-5 do TIS-10) te osobe zadužene za izvršavanje pojedinog zadataka koje su prikazane slikom profila tih osoba. Osim toga, kraj slika izvršitelja stoji i brojka koja predstavlja vremensku procjenu predviđenu za izvršavanje navedenog zadatka. U detaljima svakog zadatka se mogu vidjeti atributi kao status, prioriteti, komponente, oznake, verzija na koju zadatak utječe itd. te osobu zaduženu za izvršenje zadataka i osobu kojoj se izvještava. Ono što u detaljima zadatka nedostaje, makar nije nužna komponenta, je opis zadatka te opis testova kojim će se dokazati da je zadatak uistinu gotov.

[30] eng. *spreadsheet*

Što je zadatak deskriptivniji, to će biti razumljiviji zaduženoj osobi, a za dodavanje opisa zadacima je zadužen Vlasnik proizvoda.

4.1.3.5. Popis stavki za Sprint

Popis stavki za Sprint je skup stavki s Popisa za proizvod koji je odabran za Sprint, zajedno s planom za isporuku Inkrementa proizvoda i ostvarivanjem Cilja sprinta. Popis stavki za Sprint je predviđanje Razvojnog tima koje će funkcionalnosti biti u sljedećem Inkrementu i posla kojeg je potrebno odraditi da se funkcionalnosti isporuče u "Gotov" Inkrement. Popis stavki za Sprint čini vidljivim sav posao kojeg Razvojni tim identificira kao nužan kako bi se dosegnuo Cilj sprinta. Kako bi se osiguralo kontinuirano poboljšavanje, Popis stavki za Sprint sadrži barem jedno poboljšanje procesa visokog prioriteta koje je Razvojni tim identificirao u prethodnom Osvrtu na Sprint (Schwaber i Sutherland, 2017).

Cervone (2010) je mišljenja da je Popis stavki za Sprint isključivo kreiran od strane članova Scrum tima (dok na Popisu stavki za proizvod sudjeluju i ostali projektni dionici). Nadalje, Cervone navodi da se u idealnim uvjetima, Popis stavki za Sprint ažurira svaki dan te da ne bi smio sadržavati više od 300 zadataka. Prema Cervoneu, tim će trebati raščlaniti zadatak u manje zadatke, ako se pretpostavi da bi za zadatak trebalo više od 16 sati posla. Također, u nekim situacijama će tim morati odlučiti o stavkama koje se trebaju dodati ili oduzeti iz sprinta, ali Cervone naglašava da bi to trebala biti jedino odluka Razvojnog tima te da na nju ne bi smio utjecati Vlasnik proizvoda.

Na slici 8 su vidljiva i tri zadatka koja čine „TIS Sprint 1“ u danom primjeru. Zadaci imaju iste atribute kao oni iz Popisa stavki za proizvod (opis, status, prioriteti, oznake itd.). Čest je slučaj da se pojedini zadaci ne stignu odraditi unutar definiranog vremena Sprinta te je, u tom slučaju, potrebno prepustiti timu odluku o tome hoće li se zadatak iz Sprinta vratiti u Popis stavki za proizvod ili će ga se na Planiranju Sprinta prenijeti u sljedeći Sprint. Naravno, odluka će se temeljiti na tome koliko je prioritetno rješavanje tog specifičnog zadatka.

4.1.3.6. Inkrement

Inkrement je zbroj svih Stavki s Popisa za proizvod koje su dovršene tijekom Sprinta i vrijednost inkremenata svih prethodnih Sprinteva. Na kraju Sprinta novi Inkrement mora biti "Gotov", što znači da mora biti u upotrebljivom stanju i mora biti sukladan timskoj definiciji

"Gotovog". Svaki Inkrement podliježe kontroli i predstavlja završen posao na kraju svakog Sprinta. Inkrement je korak naprijed prema viziji ili cilju i mora biti u upotrebljivom stanju, bez obzira je li Vlasnik proizvoda odlučio pustiti Inkrement u rad ili ne (Schwaber i Sutherland, 2017).

Veoma je važna definicija „Gotovog“[31] proizvoda kako bi svi projektni dionici imali isto razumijevanje zadatka koji je „Gotov“ i može ući u inkrementalni proizvod. Definicija „Gotovog“ znači da su svi uvjeti ili kriteriji očekivanja[32] koje softverski proizvod mora ispuniti ispunjeni i spremni za prihvaćanje od strane korisnika, kupaca ili naručitelja (Huether, 2017). Huether navodi najčešće uvijete koje definicija „Gotovog“ treba sadržavati, a to su:

- Uspješno izvršeni jedinični testovi
- Kod je pregledan
- Ispunjeni su kriteriji očekivanja
- Funkcijski testovi položeni
- Ne-funkcijski zahtjevi su ispunjeni
- Vlasnik proizvoda prihvaća korisničke priče[33]

Definicija „Gotovog“ može imati više razina kompleksnosti, pa tako može biti pisana posebno za inkrementalni proizvod isporučen u Sprintu, a može i sadržavati dodatne kriterije ako se promatra aspekt puštanja proizvoda u produkciju ili na korisničko testiranje. Dakle, definicija „Gotovog“ može varirati ovisno o okolini u kojoj će se proizvod koristiti.

4.4. Prednosti i nedostaci vođenja projekta po Scrumu

Kao što je slučaj kod tradicionalne metodologije i popularnosti Vodopadnog modela, Scrum se pokazao najpopularnijim među metodama, metodologijama i razvojnim okvirima agilnoga razvoja. Istraživanje koje je provelo CollabNet, poduzeće koje je lider u razvoju softverskih rješenja koja olakšavaju agilni menadžment (Lardinois, 2014), upravo ukazuje na popularnost Scruma. U istraživanju je ispitano 1,319 zaposlenika, od kojih je 36% zaposleno u poduzeću s

[31] eng. *definition of done*
[32] eng. *acceptance criteria*
[33] Korisničke priče (eng. *user story*) su kratak i jednostavan opis mogućnosti određene funkcionalnosti. Napisane su iz perspektive korisnika ili kupca sustava. Obično slijede jednostavan format: Kao [tip korisnika], želim da [navesti cilj] tako da [dati razlog] (Huether, 2012).

manje od 1000 zaposlenika, dok je 28% ispitanika zaposleno u poduzećima s više od 20,001 zaposlenika. Najviše ispitanika radi u sektoru informacijske tehnologije (25%), dok ih slijede poduzeća koja se bave financijskim uslugama (19%), osiguranjem (8%) itd.. Naime Scrum (54%) te hibrid Scruma i ekstremnog programiranja (10%) su najkorišteniji modeli/upravljački okviri od strane poduzeća ispitanika. Osim toga, 14% poduzeća koristi neki drugi oblik hibridnog modela. (CollabNet VersionOne, 2019).

Dakle, može se reći da je Scrum korišten u poduzećima od gotovo dvije trećine ispitanika te da su tim poduzećima prednosti koje uvođenje Scruma donosi u organizaciju svakako bile vrijedne promjene.

4.4.1. Prednosti primjene Scrum razvojnog okvira

Ionel (2008) iznosi kako su tradicionalne metodologije razvoja dizajnirane samo kako bi odgovorile na nepredvidivosti internog okruženja tj. razvojnog okruženja, na početku ciklusa poboljšanja. Pristupi poput Boehmovog spiralnog modela (Boehm, 1996, navedneno u Ionel, 2008) i njegovih varijacija su i dalje ograničene u mogućnostima da odgovore na promjene zahtjeva jednom kada je projekt započeo.

Scrum metodologija[34] s druge strane, je dizajnirana kako bi bila fleksibilna kroz čitav životni ciklus projekta. Pruža mehanizme kontrole u planiranju izdavanja, a kasnije i za upravljanje ostalim varijablama kako projekt napreduje. To dozvoljava poduzećima da modificiraju projekt i isporučive proizvode u bilo kojem trenutku, isporučujući tako najprikladniju verziju (Ionel, 2008).

Prema Milleru (2018) još neki benefiti korištenja Scruma su:

- Bolja kvalitetea proizvoda – ako su kupci i ostale zainteresirane strane uključene u Preglede Sprinta i kontinuirano daju povratne informacije, krajnji proizvod će bolje udovoljavati njihovim potrebama.
- Brži povrat na investiciju – zbog činjenice da će kupci moći koristiti isporučene proizvode ranije u životnom ciklusu razvoja proizvoda.

[34] Ionel, (2008) Scrum naziva metodologijom kao što je nazvana i u brojnoj drugoj literaturi. Tek ju od nedavno autori kao što su Verheyen (2013) i Schwaber i Sutherland (2017) nazivaju razvojnim okvirom iz razloga objašnjenih u prethodnom poglavlju.

- Veća kontrola i reducirani rizik – proizlazi iz jasne podjele posla, jer u Scrumu svi od Razvojnog tima do Scrum mastera znaju točno koji je njihov posao i kada treba biti završen. Također, timovi komuniciraju i surađuju na dnevnoj bazi čime se ublažavaju potencijalne krize.
- Povećano zadovoljstvo kupaca – proizlazi iz činjenice da je primjenom Scruma moguće kupcima isporučivati dodanu vrijednost svakih dva tjedna te iz njihove uključenosti tijekom čitavog procesa razvoja proizvoda.

Također, Ionel (2008) iznosi kako Scrum daje više slobode programerima, tako da se mogu fokusirati na razvijanje inovativnih riješenja tijekom projekta, u kojem su krivulja učenja i promjene u okruženju već uzete u obzir. Mali razvojni timovi mogu dijeliti tacitna znanja vezana uz razvojne prakse, ta metodologija pruža iznimno dobro okruženje za učenje svima uključenima u projekt.

4.4.2. Nedostaci primjene Scrum razvojnog okvira

Jedna od potencijalnih slabosti Scruma istaknuta u literaturi (Highsmith i Cockburn, 2001 navedeno u Ionel, 2008) je činjenica da, kada se projekt razvija za vanjskog klijenta, on mora biti značajno uključen u projekt. Klijent mora biti u mogućnosti i na raspolaganju za testiranje verzija koje se isporučuju na mjesečnoj bazi (pa čak i dvotjednoj) te kontinuirano sugerirati nove ili modificirane funkcionalnosti.

U projektima koji koriste Scrum, vizija klijenta jako utječe na razvoj. Highsmith i Cockburn (2001 navedeno u Ionel, 2008) tvrde da u slučaju da klijent nema jasno definiran smjer razvoja proizvod, ne može ga imati ni tim. Zbog toga se na kraju konačni proizvod može značajno razlikovati od očekivanog. Stoga, prema Ionelu, jedna od najvećih prednosti koje Scrum donosi na projekte – uključenost klijenta u procese, može biti i njegova najveća mana.

Nedostatak može biti i veličina projektnog tima koja bi trebala biti, kako je već navedeno, između tri i devet članova. Ionel (2008) iznosi kako postoji način da se Scrum primijeni i na veće projekte, ali nije jednostavan za implementirati.

Još jedan potencijalni nedostatak kojeg Ionel (2008) navodi je relativno slaba vidljivost nad projektom izvan Sprinteva. Drugim riječima, teško je procijeniti koliko dugo će projekt trajati

i koliko će koštati, pogotovo u projektima u kojima se izvođači traže preko javnih natječaja ili sličnih oblika ponuda.

Postoji još mnogo prijetnji koje proizlaze iz principa kojima se vodi Scrum, kao i iz njihovog pogrešnog tumačenja ili primjene. Scrum se kao upravljački okvir sastoji od uloga i pravila koja pomažu u provedbi i facilitaciji projekta te bi se dane uloge i pravila trebali slijediti kao što su dani, a ne značajno modificirati. Postoje slučajevi u kojima poduzeća koriste nepotpun oblik Scruma te iz svojih procesa izbacuju neke događaje i ceremonije koje su ključne sastavnice Scruma. Tako nastaju hibridni modeli kojima je još teže identificirati uska grla jer nemaju jedinstven set pravila koja se slijede u razvoju i vođenju projekta.

5. STUDIJA SLUČAJA: KOMPARATIVNA ANALIZA TRADICIONALNO I AGILNO VOĐENIH PROJEKATA

Kako bi se dani teorijski okvir, opisan u dosadašnjem dijelu rada, povezao s poslovnom praksom, provest će se komparativna analiza dvaju projekata razvoja softverskih proizvoda od kojih je jedan projekt vođen po najpopularnijem modelu tradicionalne metodologije tj. vodopadnom modelu, dok je drugi projekt vođen najpopularnijim upravljačkim okvirom u sklopu agilnog razvoja tj. Scrumu. Više informacija o samim projektima izneseno je u poglavlju rezultati studije.

5.1. Općenito o analiziranim projektima

Analizirani projekti vođeni su od strane jedne od najuspješnijih hrvatskih agencija za izradu softverskih proizvoda, specijaliziranom za izradu mobilnih aplikacija za Android i iOS platforme. Prema broju zaposlenika, poduzeće spada u srednje subjekte malog gospodarstva te ima više od 250 zaposlenika.

Osim kontinuiranog i stabilnog rasta, poduzeće je nekoliko puta svrstavano u najbolje poslodavce u Republici Hrvatskoj. Agencijski poslovni model upućuje na činjenicu da poduzeće prema zahtjevima klijenta izvodi projekte razvoja softverskih proizvoda. Klijenti poduzeća su podjednako prisutni i na domaćim i na stranim tržištima. S obzirom na potpisane ugovore o tajnosti, autor rada ne smije eksplicitno iznositi imena projekta koja će biti promatrana pa je u skladu s time odlučeno ne iznositi niti ime poduzeća.

5.2. Ciljevi i hipoteze studije

Cilj studije je opisati i usporediti projektne karakteristike dvaju projekata vođenih najpopularnijim metodama tradicionalnog i agilnog načina razvoja softverskih proizvoda. Osim usporedbe, cilj je i odgovoriti na pitanje kako su opseg, vrijeme, budžet projekta i drugi faktori utjecali na odabir metodologije tj. načina vođenja projekta razvoja softverskih proizvoda. Osim toga cilj je i istražiti kako uspostavljeni procesi djeluju na sveopće zadovoljstvo članova projektnih timova uvjetima rada i napretkom projekta.

5.3. Metodologija studije slučaja

Kako bi se utvrdili ciljevi i hipoteze studije, provedeno je kvalitativno istraživanje koje je podijeljeno na dva dijela. U prvom dijelu, kroz dubinske intervjue s projektnim menadžerom zaduženim za oba projekta (jedan vođen agilno, a drugi tradicionalno) će se pokušati odgovoriti na pitanje kako su opseg, vrijeme, budžet projekta i drugi faktori utjecali na odabir metodologije tj. načina vođenja projekta razvoja softverskih proizvoda. S druge strane, iz anketnih upitnika koji su postavljeni članovima projektnih timova (dizajneri, programeri, testeri, Vlasnici proizvoda i sl.) će se opisati zadovoljstvo projektnim procesima i njihov utjecaj na radne uvijete. Anketni upitnici sastoje se od 4 pitanja na koja su odgovori ponuđeni intervalnim skalama.

5.4. Rezultati studije

Intervjuirani projektni menadžer vodi tim projektnih menadžera i nadgleda agencijski portfolio proizvoda u spomenutoj agenciji. Otkako je postao voditeljem tima, manje se bavi direktnim radom na projektima, a više se bavi nadzorom i praćenjem projekata za koje su zaduženi drugi projektni menadžeri. Transkript razgovora bit će priložen radu, a u ovom dijelu će se iznijeti najosnovniji zaključci vezani za temu istraživanja.

5.4.1. Analiza projekta A: razvoj aplikacije za mobilno bankarstvo razvijane vodopadnim modelom

Naručitelj projekta A bankarska je institucija koja je klijent poduzeća izvršitelja projekta već duži niz godina. S obzirom na činjenicu da su aplikacije mobilnih bankarstava postale „digitalne slike“ banaka, postale su i jednim od najvažnijih alata za stjecanje konkurentskih prednosti kojima se banke natječu na tržištu kojeg karakterizira velika lojalnost korisnika te malene promjene tržišnih udjela.

5.4.1.1. Model razvoja aplikacije Projekta A i projektne karakteristike

Ono što će se u sklopu ove analize promatrati jest metodologija tj. model kojim se proces razvoja pojedine aplikacije odvijao u spomenutom poduzeću. U procesu razvoja Projekta A bio je korišten vodopadni model. Razlog za primjenu vodopadnog modela bit će ispitan kroz intervju s projektnim menadžerom zaduženim za uspostavu procesa i vođenje Projekta A.

Članovi tima i njihov broj se tijekom godina mijenjao, a u trenutku ispitivanja, razvojni tim sastojao se od dva iOS programera, dva Android programera, softver testera te projektnog menadžera.

5.4.1.2. Rezultati dubinskog intervjua s projektnim menadžerom

Organizacijska struktura klijentskog poduzeća

Projekt A vodio se za poduzeće kojeg odlikuje velika korporativna struktura čija domena nije usko vezana uz razvoj softvera. Organizacijska struktura je funkcijska što ispitanik odmah povezuje s tim da je vodopadni model logičniji izbor pri definiranju projektnih aktivnosti. Ispitanik smatra kako je interna organizacija klijenta jedan od najvažnijih faktora prilikom odabira metodologije kojom će se projekt voditi i navodi kako je, prema njegovom mišljenju, teško primijeniti Scrum u velikim korporacijama jer način njihove organizacije i procesa donošenja odluka svakako ne trpi brze promjene i prilagodbe koje diktira tržište. Iz tog razloga je vodopadni model još uvijek prikladniji za odabir, ali neki primjeri iz prakse pokazuju da i Scrum može funkcionirati, ali on zahtjeva internu reorganizaciju i znatnio veće napore nego što je to slučaj u manjim poduzećima.

Budžetiranje projekta

Kod vodopadnog modela naplata se može vršiti po završetku čitavog projekta ili ona može biti podijeljena po završetcima pojedinih fazi u ciklusu razvoja. U slučaju projekta A, budžet je definiran prije početka provedbe projekta, a naplata se vršila na kraju tj. nakon što je finalni proizvod isporučen klijentu.

Proces razvoja proizvoda

Proces razvoja proizvoda u projektu A pratio je slijed prikazan u vodopadnom modelu koji je moguće vidjeti na slici 5. Ispitanik je naveo sljedeće komentare koji opisuju svaku fazu u razvoju.

1. Analiza zahtjeva

„Projektna specifikacija je bila poprilično detaljno raspisana već prije samog početka projekta te su se znale ključne karakteristike koje proizvod mora imati jednom kada izađe na tržište. Tome je pomoglo i istraživanje tržišta te postojećih konkurentskih proizvoda.” Iako je

projektna specifikacija bila detaljno raspisana, te su ključne funkcionalnosti aplikacije bile dobro opisane i poznate, ispitanik je u kasnijim pitanjima ipak naglasio da bi inzistirao na još detaljnijoj specifikaciji da taj projekt kreće iznova jer bi bilo lakše predvidjeti neke okolnosti koje su nastale tijekom razvoja i utjecale na vremenske rokove.

2. Dizajn

„Iako naša agencija pruža usluge dizajna i razvoja softverskih proizvoda, klijenti u nekim slučajevima znaju imati već gotov dizajn te je naš zadatak u tom slučaju samo razvoj i testiranje gotovog proizvoda. U ovom slučaju smo mi radili i dizajn tako da su faze „školski" slijedile kako to vodopadni model nalaže. Prvo smo krenuli s dizajnom proizvoda. Obično se u početku rade tzv. „wireframeovi" koji daju okvirnu sliku kako će mobilna aplikacija izgledati, ali sadržavaju samo ključne elemente korisničkog sučelja i navigacije na njemu. Oni se dostavljaju klijentu te se čeka njihova povratna informacija oko eventualnih iteracija. To eliminira rizik da se napravi neki značajniji posao, a klijent ne bude zadovoljan postignutim. Nakon nekoliko iteracija wireframeova su oni potvrđeni te je započet dizajn koji je uključivao crtanje svih ekrana aplikacije te je ta faza zajedno s wireframingom trajala oko 2 mjeseca." Za čitav dizajn aplikacije bila je zadužena jedna osoba.

3. Razvoj

„Budući da je riječ o jako složenom projektu te da se u proces moraju uključivati i brojne treće strane koje pružaju određene specifične servise neophodne za funkcioniranje takve aplikacije, razvoj je bio iznimno kompleksan i dugotrajan, ako se dobro sjećam trajao je između šest i osam mjeseci. U fazi razvoja najaktivniji su bili iOS i Android inženjeri (jedna osoba po platformi) dok je angažman dizajnera, testera i projektnog menadžera bio manji, ovisno o potrebitosti situacije."
Razvoj proizvoda je faza koja traje najduže što potvrđuje i komentar projektnog menadžera. U ovom slučaju, razvoj je iznosio oko 80% ukupnog vremena utrošenog na sve faze životnog ciklusa.

4. Testiranje

„Što se testiranja tiče, ono se vršilo i periodički kroz razvoj budući da je riječ o dosta kompleksnom proizvodu, ali najveći naglasak je bio ipak kada je proizvod bio u potpunosti dovršen i spreman za produkciju". Iako periodičko testiranje nije karakteristično za

tradicionalnu metodologiju i vodopadni model, u ovom slučaju je bilo potrebno zbog veće kompleksnosti proizvoda i rizika koji bi proizašao iz nedovoljno testiranog proizvoda. Zadnja faza testiranja je trajala dva tjedna prije nego što su se produkcijske verzije, spremne za stavljanje na Appleove i Googleove trgovine, dostavile klijentu.

5. Održavanje

„Nakon što je proizvod bio gotov i istestiran, dogovoren je mjesečni budžet za održavanje i eventualne popravke na aplikaciji." Faza održavanja je trajala oko šest mjeseci te je nakon nje, iz zahtjeva klijenta za konkretnijim doradama i novim funkcionalnostima, započet novi proces razvoja zajedno sa svim fazama vodopadnog modela.

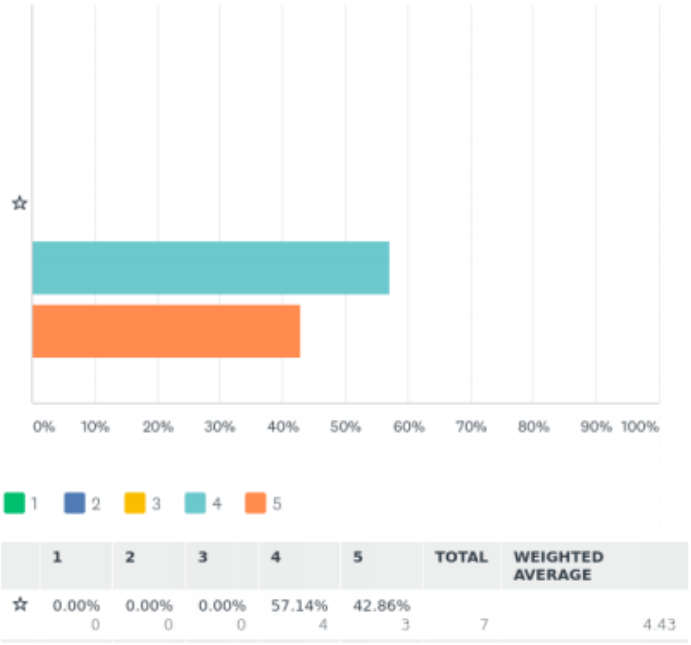
Komunikacija

Ispitanik iznosi kako su rokovi i miljokazi bili postavljeni prije započinjanja projekta, ali oni nisu bili strogo definirani.
„Rokovi su bili postavljeni za dovršavanje čitave aplikacije, ali za svaku pojedinu fazu nisu bili strogo specificirani. Kada bi određena faza bila gotova, to bi se nagovijestilo te bi se napravio određeni „status check" sastanak. Ključne osobe s klijentske strane su bile aktivno uključene u proces razvoja te im se učestalo komunicirao status, ali to nije bilo na svakodnevnoj razini niti periodički definirano kao što je to slučaj sa Scrumom."

5.4.1.3. Rezultati ankete o projektnom zadovoljstvu na projektu A

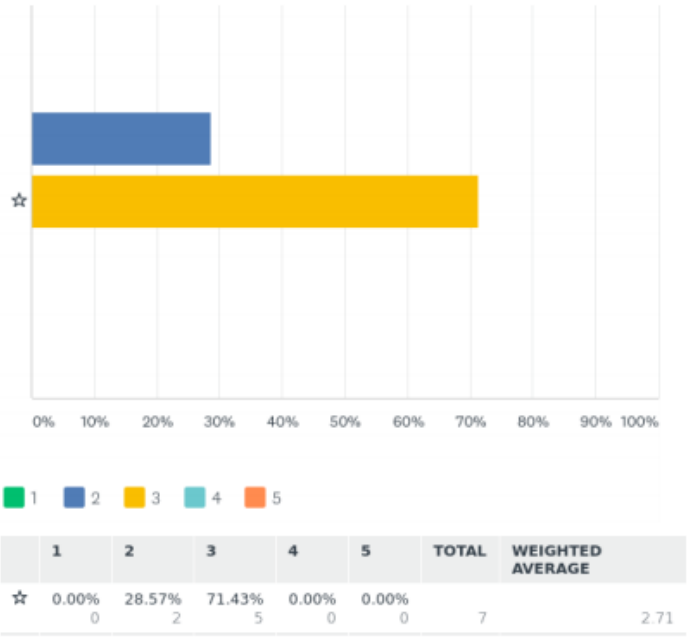
Anketom o projektnom zadovoljstvu ispitani su članovi tima koji su aktivno sudjelovali u procesu razvoja proizvoda u projektu A. Anketom je ispitano petero članova u agenciji zaduženoj za razvoj proizvoda te dvije osobe iz poduzeća naručitelja projekta. Članovi tima su dva Android inženjera, dva iOS inženjera, softver tester te dvoje specijalista iz područja poslovanja kojim se klijentsko poduzeće bavi.

Slika 9 Pitanje 1: Koliko ste zadovoljni projektom općenito? (skala od 1-5, pri tome 1= nimalo nisam zadovoljan, 5 = u potpunosti sam zadovoljan)



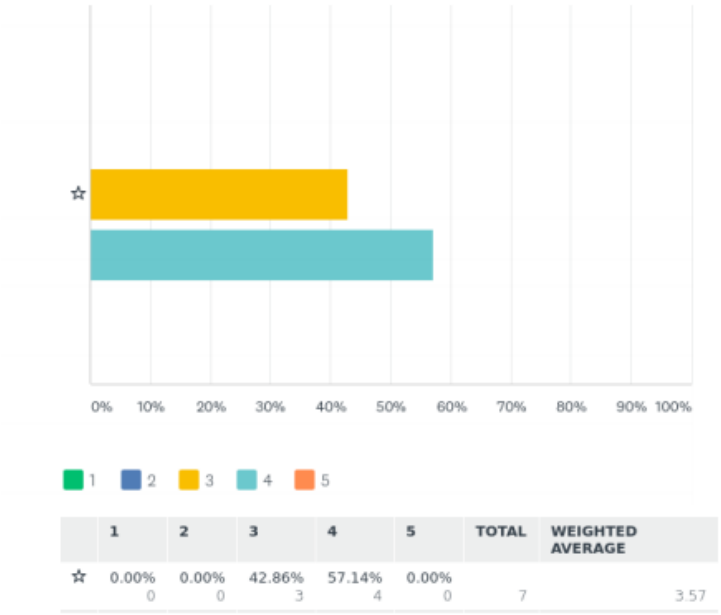
	1	2	3	4	5	TOTAL	WEIGHTED AVERAGE
☆	0.00% 0	0.00% 0	0.00% 0	57.14% 4	42.86% 3	7	4.43

Slika 10 Pitanje 2: Koliko vam je radno opterećenje na projektu? (skala od 1-5, pri tome 1= imam premalo posla, 5 = imam previše posla)



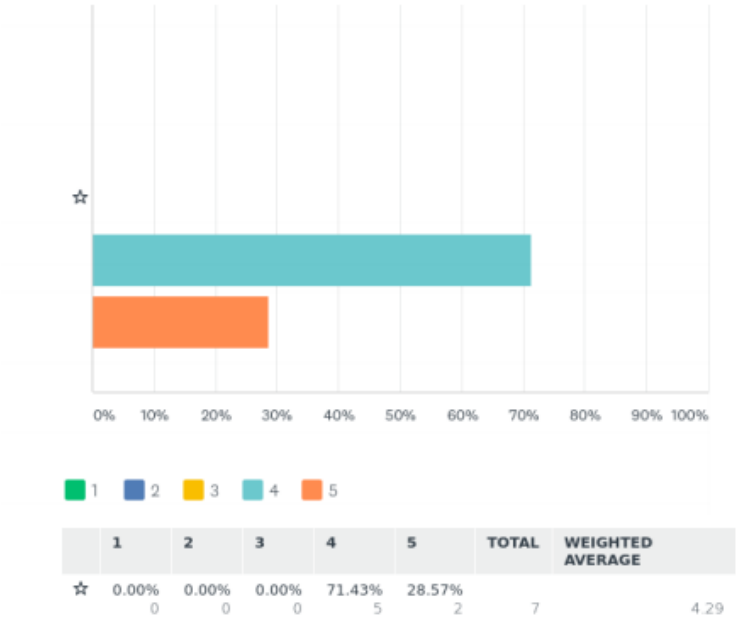
	1	2	3	4	5	TOTAL	WEIGHTED AVERAGE
☆	0.00% 0	28.57% 2	71.43% 5	0.00% 0	0.00% 0	7	2.71

Slika 11 Pitanje 3: Koliko dobro je projekt organiziran? (skala od 1-5, pri tome 1= kaotično, 5 = savršeno)



	1	2	3	4	5	TOTAL	WEIGHTED AVERAGE
☆	0.00% 0	0.00% 0	42.86% 3	57.14% 4	0.00% 0	7	3.57

Slika 12 Pitanje 4: Jesu li sve informacije od kritične važnosti jasno komunicirane? (skala od 1-5, pri tome 1= gotovo ništa nije komunicirano, 5 = uvijek sam informiran o važnim temama)



	1	2	3	4	5	TOTAL	WEIGHTED AVERAGE
☆	0.00% 0	0.00% 0	0.00% 0	71.43% 5	28.57% 2	7	4.29

5.4.2. Analiza projekta B: razvoj mobilnog novčanika za kriptovalute razvijanog prema Scrum okviru

Naručitelj Projekta B je poduzeće koje je vlasnik internetske platforme koja služi za primanje, slanje, trgovinu i upravljanje kriptovalutama. S obzirom na potrebu za povećanjem tržišnog udjela te pružanju šireg spektra usluga svojim korisnicima, poduzeće se odlučilo za razvoj Android i iOS mobilnih aplikacija koje će služiti kao novčanik za pohranu valuta, ali i platforma za njihovu kupovinu i razmjenu. Projekt je pokrenut sredinom 2019. godine, a prve verzije aplikacije će biti dostupne korisnicima na korištenje u 2020. godini.

5.4.2.1. Model razvoja aplikacije Projekta B i projektne karakteristike

Scrum upravljački okvir poslužio je prilikom uspostave i provođenja projektnih procesa u Projektu B. Imajući na umu ključne sastavnice, prilikom pokretanja projekta bilo je potrebno utvrditi uloge članova te odabrati projektne članove. Projektni tim se sastoji od dvojice Vlasnika proizvoda, Scrum mastera, te Razvojnog tima koji se inicijalno sastojao od tri programera za iOS platformu, tri programera za Android platformu, te jednog softver testera tj. od ukupno deset članova.

5.4.2.2. Rezultati dubinskog intervjua s projektnim menadžerom

Organizacijska struktura klijentskog poduzeća

Klijentsko poduzeće koje je naručitelj projekta B relativno je mlado poduzeće koje je osnovano tek prije pet godina. Broji ispod 50 zaposlenika te ga karakterizira jednostavna, izrazito neformalna organizacijska struktura, pogotovo ako ju usporedimo sa strukturom klijentskog poduzeća u projektu A. Zaposlenici se dijele u timove, a uprava ima direktnu superviziju nad gotovo svim procesima. S obzirom na to da je poduzeće mlado te ljudi nisu usko specijalizirani, već im domena obuhvaća veći spektar aktivnosti, ispitanik povlači paralelu s time da takve karakteristike olakšavaju primjenu Scruma. Još jednom je bitno naglasiti kako ispitanik navodi upravo organizacijsku strukturu klijenta kao jednu od najvažnijih determinanti koje će utjecati na odabir metodologije vođenja projekta.

Budžetiranje projekta

Iz razgovora s projektnim menadžerom, sljedeći dio se odnosi na budžetiranje projekta vođenog Scrumom:

„Kod Scruma, pošto ne poznajemo specifikaciju, tj. specifikacija se definira kroz iterativno izvođenje, budžet se definira resursno – koliko čovjek/sati/dana će se utrošiti u određenom periodu. Također bih rekao da iskustvo pokazuje da su waterfall projekti u software developmentu skuplji. Scrum je u tom smislu jeftiniji jer omogućava brži „time to market““ Tako se primjerice izračuna ukupno ljudstvo potrebno da bi se odradio posao za jedan sprint te se ukupan broj sati njihovog pomnoži cijenom sati koji se naplaćuje. Dobiveni iznos predstavlja iznos koji će se fakturirati po završetku Sprinta.

Proces razvoja proizvoda

Projekt B je u trenutku preuzimanja od strane agencije već imao specificiran dizajn proizvoda, ali on nije bio kompletan te je agencija imala zadatak napraviti i neke manje dorade. Ispitanik ovako objašnjava izbor načina vođenja projekta B:

„Zbog činjenice da specifikacija nije bila najdetaljnije raspisana te zbog činjenice da je riječ o mlađem poduzeću za kojeg smo procijenili da bi moglo biti relativno responzivno u čitavom procesu razvoja i komunikacije, odlučili smo se za agilan razvoj kako bismo što prije izašli s proizvodom van te počeli prikupljati povratne informaciji od korisnika.“

1. Sprintevi

Jednom kada je projekt bio uspostavljan, započeti su dvotjedni sprintevi. Duljina trajanja sprinta od dva tjedna se činila kao idealna s obzirom na klijentova očekivanja i zahtjeve za inkrementima proizvoda.

2. Planiranje Sprinta

Procese planiranja Sprinta, ispitanik opisuje sljedećim riječima:

„Prije započinjanja svakog Sprinta odradio bi se sastanak na kojem bi bili prisutni Vlasnik proizvoda i čitav Razvojni tim. Ne njemu bi se diskutiralo o zadacima iz Popisa stavki za proizvod te bi se procijenjivalo vrijeme potrebno za njihovo dovršavanje ako to nije bilo odrađeno ne nekom od sastanaka prije. Kod Planiranja Sprinta svaki član Razvojnog tima

procjenjuje vrijeme koje mu je slobodno za rad na projektu, a iz njega se isključuju sve interne obaveze koje pojedinac ima prema poduzeću u kojem radi, pauze za ručak i slično. Kada se dobije vrijeme koje čitav tim ima slobodno za nadolazeći Sprint, to vrijeme se alocira na procijenjene zadatke te se time pokušava izbjeći situacija u kojoj članovi imaju previše ili premalo posla što zna biti čest slučaj kod vodopadnog modela.“

3. Dnevni sastanci

Ispitanik navodi sljedeće:

„Daily sastanke nismo imali svaki dan kako to nalaže Scrum upravljački okvir, već tri puta tjedno. Pokušavali smo se držati petnaestominutnog roka koliko bi on trebao trajati te smo uglavnom u tome i uspijevali. Na tim sastancima bi bio prisutan čitav razvojni tim s naše strane te dva vlasnika proizvoda sa strane klijenta. Pratio se format koji nalaže Scrum te bi na tim sastancima svatko iz tima dao status info o tome na čemu je radio dan prije, na čemu planira raditi taj dan te blokira li ga nešto u radu.“ Iz ovog primjera je vidljivo kako modifikacija Scruma (Dnevni sastanci se održavaju tri puta tjedno umjesto pet puta) prema potrebama tima nekada može imati smisla te rezultirati povoljnim ishodom.

4. Pregled i Osvrt na Sprint

Ispitanik ovako opisuje navedene sastanke:

„Po završetku sprinta bi imali sastanke koji se zovu Pregled i Osvrt na Sprint. Iste stranke su bile prisutne na njemu kao i na daily sastancima te bi na tom sastanku komentirali napredak napravljen u proteklom Sprintu. Održavala se i demonstratura razvijenih funkcionalnosti tijekom Sprinta te se osvrtalo na pozitivne i negativne prakse tijekom njegovog trajanja. Pod demonstraturom mislim na to da bi članovi razvojnog tima kroz testne aplikacije pokazivali klijentima koje su se funkcionalnosti razvile tijekom Sprinta. Klijenti su tako dobili prve opipljive verzije aplikacije već nakon mjesec dana razvoja. One su dakako bile tek u začetku, ali su se mogle vidjeti i istestirati prve funkcionalnosti koje su se krenule razvijati.“
Važnost Osvrta na Sprint upravo je u tome da klijent što prije dobije prvu verziju proizvoda kako bi se što prije mogle vršiti prilagodbe novim zahtjevima.

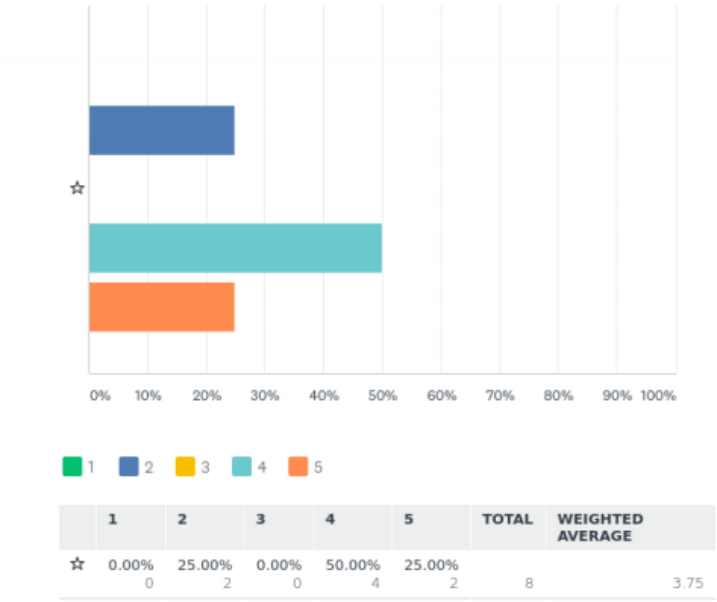
Komunikacija

Iz opisanih procesa, jasno je da je komunikacija među članovima projektnog tima učestala te da ju Scrumom propisani događaji potiču na svakodnevnoj bazi kroz dnevne sastanke te pri započinjanju i završavanju Sprinteva kroz Planiranje Sprinta te Pregled i Osvrt na Sprint.

5.4.2.3. Rezultati ankete o projektnom zadovoljstvu na projektu B

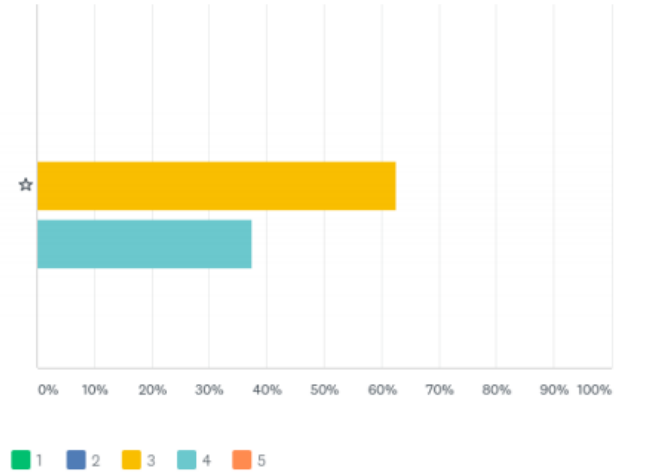
Anketom o projektnom zadovoljstvu ispitani su članovi tima koji su aktivno sudjelovali u procesu razvoja proizvoda u projektu B. Anketom je ispitano šestero članova u agenciji zaduženoj za razvoj proizvoda te dvije osobe iz poduzeća naručitelja projekta. Ispitanici obnašaju sljedeće funkcije: iOS inženjer (3), Android inženjer (3), Vlasnik proizvoda (2)

Slika 13 Pitanje 1: Koliko ste zadovoljni projektom općenito? (skala od 1-5, pri tome 1= nimalo nisam zadovoljan, 5 = u potpunosti sam zadovoljan)



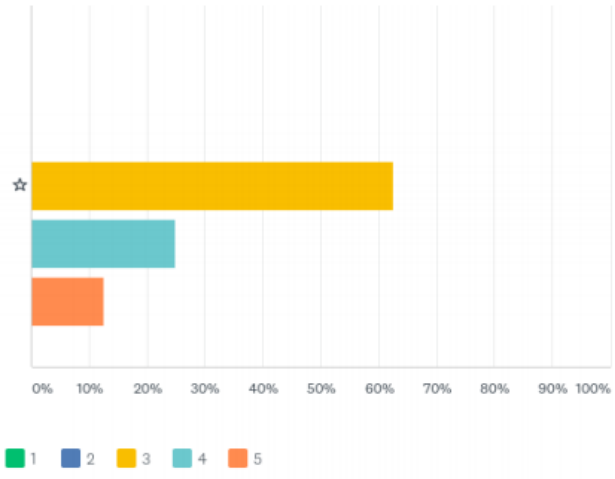
	1	2	3	4	5	TOTAL	WEIGHTED AVERAGE
☆	0.00% 0	25.00% 2	0.00% 0	50.00% 4	25.00% 2	8	3.75

Slika 14 Pitanje 2: Koliko vam je radno opterećenje na projektu? (skala od 1-5, pri tome 1= imam premalo posla, 5 = imam previše posla)



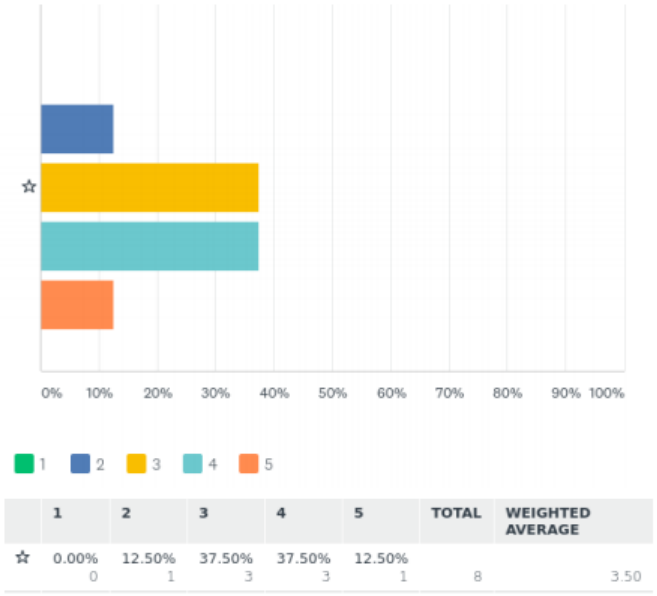
	1	2	3	4	5	TOTAL	WEIGHTED AVERAGE
☆	0.00% 0	0.00% 0	62.50% 5	37.50% 3	0.00% 0	8	3.38

Slika 15 Pitanje 3: Koliko dobro je projekt organiziran? (skala od 1-5, pri tome 1= kaotično, 5 = savršeno)



	1	2	3	4	5	TOTAL	WEIGHTED AVERAGE
☆	0.00% 0	0.00% 0	57.14% 4	28.57% 2	14.29% 1	7	3.57

Slika 16 Pitanje 4: Jesu li sve informacije od kritične važnosti jasno komunicirane? (skala od 1-5, pri tome 1= gotovo ništa nije komunicirano, 5 = uvijek sam informiran o važnim temama)



	1	2	3	4	5	TOTAL	WEIGHTED AVERAGE
☆	0.00% 0	12.50% 1	37.50% 3	37.50% 3	12.50% 1	8	3.50

5.5. Zaključak studije

S obzirom na dva navedena primjera projekata te na komentare iz razgovora s projektnim menadžerom, moguće je zaključiti da su neke projektne karakteristike ključne pri odabiru metodologije kojom će se projekt voditi, a one bi bile:

- Stupanj poznavanja projektne specifikacije
- Organizacijska struktura poduzeća
- Način budžetiranja projekta i planiranja projektnih troškova

Svakako da postoji još nekolicina parametara koji će pomoći u donošenju odluke te da se oni svakako razlikuju od projekta do projekta, no u ovim primjerima dvaju projekata razvoja softverskih proizvoda, veličina tima, vrijeme trajanja projekta te opseg projekta nisu bili izneseni kao ključni faktori prilikom formiranja odluke. Također, ono što je važno usporediti jesu težinski prosjeci odgovora ispitanika dobivenih iz anketnih upitnika, a oni su prikazani u tablici 5.

Tablica 4 Usporedba rezultata anketa o projektnom zadovoljstvu.

Čimbenik	Težinski prosjek projekta A	Težinski prosjek projekta B
Zadovoljstvo	4.43	3.75
Radno opterećenje	2.71	3.38
Organizacija	3.57	3.57
Informiranosti	4.29	3.5

Usporedbom dobivenih rezultata možemo vidjeti da je u svakom segmentu projekt A (vodopadni model) dobio bolju ocjenu osim u organizaciji projekta, koja je igrom slučaja dobila identičnu težinsku ocjenu. Ako promatramo faktor radnog opterećenja, ocjena 3 (sredina) bila bi optimum, što znači da je bolje ocijenjen projekt A jer manje udaljen od tog optimuma. Naravno da ne treba suditi uspješnost metodologije na rezultatima jedne ovako jednostavno postavljene ankete i na primjeru samo dva projekta. Veću pozornost treba obratiti na činjenicu da, ako je projektna metodologija ispravno odabrana te su ključni faktori razmotreni prije donošenja takve odluke, se može postići zadovoljstvo članova tima neovisno o odabranom načinu vođenja projekta. Također, metodologije se ne bi trebale odabirati proizvoljno, bilo zbog njihovih popularnosti ili drugih faktora, već isključivo kao predmet detaljne analize projektnih zahtjeva i karakteristika.

6. ZAKLJUČAK

Projektni menadžment je disciplina menadžmenta čije se karakteristike znatno razlikuju s obzirom na karakteristike industrije unutar kojih se promatraju pojedini projekti. Iako projekt gradnje kuće i projekt razvoja mobilne aplikacije na prvi pogled nemaju mnogo zajedničkih dodirnih točaka, ipak postoje univerzalna područja znanja u projektnom menadžmentu koja olakšavaju rad osobama zaduženim za uspostavu projektnih procesa i njihovo izvršenje. S vremenom su se ta znanja formalizirala i grupirala unutar naziva kao što su projektne metodologije. Metodologije u projektnom menadžmentu se obično definiraju kao skupine metoda, tehnika, procedura, pravila, predložaka i najboljih praksi koje se koriste na projektu (Šundak, 2014). Tako je dugi niz godina, tradicionalna metodologija te najpopularniji model proizašao iz pravila propisanih tradicionalnom metodologijom – vodopadni model, bio uobičajeni izbor prilikom postavljanja projektnih procesa. Vodopadni model, baš kao što mu to i samo ime govori, karakterizira kaskadni redoslijed faza u životnom ciklusu razvoja proizvoda. Drugim riječima, u njemu su točno propisane faze razvoja (analiza projektnih zahtjeva, dizajn, razvoj, testiranje, održavanje). Kraj jedne faze signalizira početak iduće te one se u pravilu ne bi smjele preklapati. S obzirom na značajke softverskih proizvoda kao digitalnih proizvoda, moderne metodologije i načini razvoja softvera iskorištavaju principe agilnog i iterativnog načina kako bi se maksimiziralo udovoljavanje korisničkim zahtjevima te omogućilo što brže i češće puštanje proizvoda na tržište. Jedan od takvih načina je i Scrum upravljački okvir. Scrum ne sadrži strogi niz metoda, tehnika, procedura, pravila, predložaka i najboljih praksi već objedinjuje principe agilnog načina razvoja u okvir koji propisuje niz uloga projektnih članova, projektnih događaja i čimbenika čijom se primjenom mogu utilizirati principi iterativnog i inkrementalnog razvoja proizvoda. Na primjerima projekata analiziranih u studiji slučaja može se vidjeti kako ključni faktori kao što su organizacijska struktura naručitelja projekta, stupanj poznavanja i razvijenosti specifikacije proizvoda te plan budžetiranja mogu imati značajnu ulogu u odabiru projektne metodologije. Zaključak je da oba načina uspostave projektnih procesa mogu biti jednako uspješna i rezultirati zadovoljnim članovima projektnih timova ako su odluke o uspostavi tih procesa donesene nakon detaljne analize projektnih zahtjeva i karakteristika budućeg proizvoda.

LITERATURA

1. Almeida, F. (2017). Challenges in migration from waterfall to agile environments. *World Journal of Computer Application and Technology*, 5(3), 39-49.
2. Awad, M. A. (2005). A comparison between agile and traditional software development methodologies. *University of Western Australia*, 30.
3. Awad, M. A. (2005). A comparison between agile and traditional software development methodologies. *University of Western Australia*, 30.
4. Bassil, Y. (2012). A simulation model for the waterfall software development life cycle. *International Journal of Engineering & Technology*, 2(5).
5. Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, (5), 61-72.
6. Cervone, H. F. (2011). Understanding agile project management methods using Scrum. *OCLC Systems & Services: International digital library perspectives*, 27(1), 18-22.
7. Charvat, J. (2003*). Project management methodologies: selecting, implementing, and supporting methodologies and processes for projects*. Hoboken, New Yersey: John Wiley & Sons.
8. Conforto, E. C., Salum, F., Amaral, D. C., Da Silva, S. L., De Almeida, L. F. M. (2014). Can agile project management be adopted by industries other than software development?. *Project Management Journal*, 45(3), 21-34.
9. Čubranić, D., Kaluža, M., & Novak, J. (2013). Standardne metode u funkciji razvoja softvera u Republici Hrvatskoj. *Zbornik Veleučilišta u Rijeci*, 1(1), 239-256.
10. Demir, K. A. (2009). A Survey on Challenges of Software Project Management. *Software engineering research and practice*, *2009*, 579-585.
11. Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *The Journal of Systems and Software*. 85 (6), 1213-1221.
12. Dybå, T., Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and software technology*, 50(9-10), 833-859.

13. Ebert, L. (2002). Successful project management for software product and information system development. *Project Management Institute Annual Seminars & Symposioum, San Antonio, TX. Newtown Square*
14. Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. *Computer*, *34*(9), 122.
15. Javanmard, M., Alian, M. (2015). Comparison between Agile and Traditional software development methodologies. *Cumhuriyet Üniversitesi Fen-Edebiyat Fakültesi Fen Bilimleri Dergisi*, *36*(3), 1386-1394.
16. Kannan, V., Jhajharia, S., Verma, S. (2014). Agile vs waterfall: A Comparative Analysis. *International Journal of Science, Engineering and Technology Research (IJSETR),* 3(10), 2680-2686.
17. Kerzner, H. (2017*). Project management: a systems approach to planning, scheduling, and controlling*. 7. Izd. Ohio: John Wiley & Sons.
18. Kumar, G., Bhatia, P. K. (2014). Comparative analysis of software engineering models from traditional to modern methodologies. *Fourth International Conference on Advanced Computing & Communication Technologies*. Rohtak: IEEE.
19. Leau, Y. B., Loo, W. K., Tham, W. Y., Tan, S. F. (2012). Software development life cycle AGILE vs traditional approaches. In *International Conference on Information and Network Technology,* 37(1), 162-167.
20. Liović, D. (2016), Outsourcing – rizična ušteda*?, Book of Proceedings of Singidunum University International Scientific Conference Risks in Contemporary Business.* 223-230.
21. Manger, R. (2016). *Softversko inženjerstvo*. Zagreb: Element
22. Martin, R. C. (2002). *Agile software development: principles, patterns, and practices.* New Yersey: Prentice Hall.
23. Matharu, G. S., Mishra, A., Singh, H., Upadhyay, P. (2015). Empirical study of agile software development methodologies: A comparative analysis. *ACM SIGSOFT Software Engineering Notes*, 40(1), 1-6.
24. Panian, Ž., Ćurko, K. (2010). *Poslovni informacijski sustavi*. Zagreb: Element.
25. Rising, L., & Janoff, N. S. (2000). The Scrum software development process for small teams. *IEEE software*, 17(4), 26-32.
26. Ruparelia, N. B. (2010). Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes*, 35(3), 8-13.

27. Schwalbe, K. (2015). *Information technology project management*. 8. izd. Boston: Cengage Learning.
28. Siegelaub, J. M. (2007). Six (yes six!) constraints: an enhanced model for project control. *PMI Global Congress 2007, Newtown Square, North America, Atlanta, GA.*
29. Stellman, A., Greene, J. (2005). *Applied software project management*. Sebastopol: O'Reilly Media.
30. Špundak, M. (2014). Mixed agile/traditional project management methodology–reality or illusion?. *Procedia-Social and Behavioral Sciences*, 119(1), 939-948.
31. Vijayasarathy, L. R., Butler, C. W. (2015). Choice of software development methodologies: Do organizational, project, and team characteristics matter?. *IEEE software*, 33(5), 86-94.
32. Vlahov, R.D. (2013). 'Projektni menadžment na hrvatski način', *Ekscentar*, 16, 116.
33. Wysocki, Robert K. (2011). *Effective software project management*. 5. izd. Indiana: John Wiley & Sons.

POPIS KORIŠTENIH IZVORA

1. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D. (2001). Proglas o metodi agilnog razvoja softvera [online]. Dostupno na http://agilemanifesto.org/iso/hr/manifesto.html [pristupano: 27.12.2019]
2. ColabNet VersionOne (2019). 13th annual state of agile report [online]. Dostupno na https://explore.versionone.com/state-of-agile/13th-annual-state-of-agile-report [pristupano 03.01.2020.]
   https://www.workfront.com/blog/the-6-project-constraints [pristupano: 28.11.2019]
3. Huether, D. (2012). What is a user story [online]. Dostupno na https://www.leadingagile.com/2012/07/user-story/ [pristupano: 29.12.2020]
4. Huether, D. (2017) Definition of done [online]. Dostupno na https://www.leadingagile.com/2017/02/definition-of-done/ [pristupano: 02.01.2020.]
5. Jira Software Support, (2019). Using your Scrum backlog [online]. Dostupno na https://confluence.atlassian.com/jirasoftwareserver079/using-your-scrum-backlog-950290616.html [pristupano: 02.01.2020.]
6. Lardinois, F. (2014). Vector Capital Makes Controlling Investment In CollabNet Developer Platform [online]. Dostupno na https://techcrunch.com/2014/01/31/vector-capital-makes-controlling-investment-in-collabnet-developer-platform/ [pristupano 03.01.2020]
7. Noyes, D. (2019). The top 20 Valuable Facebook Statistics – Updated September 2019. Zephoria [online]. Dostupno na https://zephoria.com/top-15-valuable-facebook-statistics/ [pristupano: 17.10.2019]
8. Schwaber, K. I Sutherland J. (2017). Sveobuhvatni vodič kroz Scrum: Pravila igre, [online]. Dostupno na https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Croatian.pdf#zoom=100 [pristupano: 31.12.2019]
9. Shenoy, S. (2017). The 6 Project Constraints. Workfront [online]. Dostupno na
10. Verheyen, G. (2013). Scrum: Framework, not methodology [online]. Dostupno na https://guntherverheyen.com/2013/03/21/scrum-framework-not-methodology/ [pristupano: 29.12.2019]

11. Verheyen, G. (2017). Scrum: What's in a Name?. Dzone [online]. Dostupno na https://guntherverheyen.com/2013/03/21/scrum-framework-not-methodology/ [pristupano: 29.12.2019]
12. Wykowski, T., Wykowska, J. (2018). Lessons learned: Using Scrum in non-techical teams. Agile Alliance [online]. Dostupno na https://www.agilealliance.org/resources/experience-reports/lessons-learned-using-scrum-in-non-technical-teams/ [pristupano: 29.12.2019]
13. Zapier, (2019). How do we work. Zapier [online]. Dostupno na https://zapier.com/jobs/ [pristupano: 12.11.2019]

POPIS SLIKA

POPIS TABLICA

POPIS GRAFIKONA

ŽIVOTOPIS STUDENTA

Luka Marić rođen je 04.06.1995. godine u Zagrebu, gdje je završio OŠ Trnsko, a zatim Prvu gimnaziju u Zagrebu. Nakon završene srednje škole, upisuje preddiplomski sveučilišni studij Poslovne ekonomije 2014. godine na Ekonomskom fakultetu u Zagrebu te ga završava 2018. godine. Kao student ekonomije, pronašao je interes u poduzetništvu te svoje prve ideje testira na studentskim natjecanjima kao što je Case Study Competition, na kojem je osvojio prvo mjesto 2018. godine u rješavanju poslovnog slučaja Hrvatskog Telekoma. Tim je formirao s kolegama s Ekonomskog fakulteta te Fakulteta elektrotehnike i računarstva. U 2018. godini, tražeći studentsku praksu, dolazi do ideje o kreiranju web stranice koja bi predstavljala centralizirano mjesto na kojem bi studenti tražili studentske prakse, pripravništva i ostale poslove koji bi im predstavljali prvi korak u započinjanju karijernog puta na završnim godinama formalnog obrazovanja. Svoju ideju predstavlja na StartupZG natjecanju u organizaciji Zagrebačkog inkubatora poduzetništva te osvaja prvo mjesto. Nedugo zatim osvaja još nekoliko nagrada na natjecanjima u prezentiranjima poduzetničkih ideja te sa svojim projektom osvaja inkubaciju u HUB385 i AlgebraLab inkubatorima. Trenutno je redovan student pete godine sveučilišnog diplomskog studija na smjeru Analize i poslovnog planiranja na Ekonomskom fakultetu u Zagrebu. Na petoj godini studija zapošljava se u jednoj od najpoznatijih hrvatskih agencija za razvoj softverskih proizvoda na poziciji Junior Project Manager te u manje od godine dana napreduje do pozicije Project Manager. Kao Project Manager nastavlja slijediti strast kreiranja i razvoja digitalnih proizvoda te mu teme primjene projektnih metodologija u razvoju softverskih proizvoda postaju predmetom istraživanja kako u slobodno vrijeme tako i u pisanju diplomskog rada.