IZAZOVI UPRAVLJANJA PROJEKTIMA RAZVOJA SOFTVERA PRIMJENOM SCRUM OKVIRA ZA AGILNO UPRAVLJANJE PROJEKTIMA

Radoš, Ana

Master's thesis / Diplomski rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Economics and Business / Sveučilište u Zagrebu, Ekonomski fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:148:955345

Sveučilište u Zagrebu
Ekonomski fakultet
Integrirani preddiplomski i diplomski sveučilišni studij
Poslovna ekonomija − smjer Menadžment

IZAZOVI UPRAVLJANJA PROJEKTIMA RAZVOJA SOFTVERA PRIMJENOM SCRUM OKVIRA ZA AGILNO UPRAVLJANJE PROJEKTIMA

Diplomski rad

Ana Radoš

Zagreb, rujan 2021.

Sveučilište u Zagrebu
Ekonomski fakultet
Integrirani preddiplomski i diplomski sveučilišni studij
Poslovna ekonomija − smjer Menadžment

IZAZOVI UPRAVLJANJA PROJEKTIMA RAZVOJA SOFTVERA PRIMJENOM SCRUM OKVIRA ZA AGILNO UPRAVLJANJE PROJEKTIMA

CHALLENGES OF SOFTWARE DEVELOPMENT PROJECT MANAGEMENT USING THE SCRUM FRAMEWORK FOR AGILE PROJECT MANAGEMENT

Diplomski rad

Student: Ana Radoš
JMBAG: 0067557233
Mentor: Izv. prof. dr. sc. Rebeka Danijela Vlahov Golomejić

Zagreb, rujan 2021.

ZAHVALA

Veliku zahvalu dugujem svojim roditeljima, Valeriji i Fabijanu Radošu, koji su mi omogućili sve kako bih danas bila tu gdje jesam. Veliko im hvala za konstantu podršku i razumijevanje koje su pružali, kao i strpljenje koje su pokazali tijekom moga studija. Također zahvaljujem svojoj mentorici, Rebeki Vlahov Golomejić, za njezinu konstantnu dostupnost kao i za to što me kroz ovaj rad vodila svakim korakom i na svaki moj upit spremno odgovarala.

____Ana Radoš_____
Ime i prezime studentice

IZJAVA O AKADEMSKOJ ČESTITOSTI

Izjavljujem i svojim potpisom potvrđujem da je diplomski rad 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 izvora 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.

Studentica:

U Zagrebu, 20.09.2021.

(potpis)

SAŽETAK I KLJUČNE RIJEČI NA HRVATSKOM JEZIKU

Do kraja prošlog stoljeća organizacije su počele uviđati da je primjena upravljanja projektima više stvar potrebe, a ne izbora. Iako se upravljanje projektima proširilo na gotovo sve grane gospodarstva, primjetno je kako je velik broj IT i ICT poduzeća značajno profitirao primjenjivanjem istog. Zbog sve veće popularnosti razvoja softvera u ovome inovativnom dobu iskazala se potreba za brojnim metodologijama kako bi se adekvatno upravljalo takvim kompleksnim projektima. Jedan od najpopularnijih izbora u ovim industrijama je Scrum okvir. Scrum okvir predstavlja način na koji se projekt izvodi uz određene artefakte, uloge i aktivnosti, koje u konačnici rezultiraju velikom vrijednosti za projekt. Svrha rada je ispitati izazove koji se pojavljuju prilikom korištenja Scrum okvira za razvoj softverskih projekta. Stoga, u teorijskom dijelu objašnjen je pojam agilnog upravljanja projektima, kao i njegove metodologije i okvira te je opširnije pojašnjen Scrum okvir. Također, pojašnjene su i specifičnosti upravljanja projektima razvoja softvera, točnije njegovi trendovi, odrednice i izazovi. Empirijski dio rada donosi istraživanje izazova upravljanja projektima razvoja softvera primjenom Scrum okvira za agilno upravljanje projektima u hrvatskim IT i ICT poduzećima. Istraživanje je provedeno anketom ispitujući 39 glavnih izazova sa Scrumom. Ista je upućena spomenutim poduzećima i zaprimila je po minimalno jedan odgovor (po poduzeću), od različitih relevantnih osoba unutar projektnog tima. Rezultati istraživanja pokazali su kako se, usprkos brojnim prednostima Scrum okvira, poduzeća i dalje susreću sa značajnim brojem izazova različitih spektra. Unatoč tomu organizacije i dalje biraju Scrum okvir i isporučuju uspješne softvere.

Ključne riječi: upravljanje projektima, Scrum okvir, IT i ICT industrija, softver, izazovi

SUMMARY AND KEY WORDS

By the end of the last century, organizations began to realize that the application of project management was more a matter of need rather than choice. Although project management has spread to almost all sectors of the economy, it is noticeable that a large number of IT and ICT companies have benefited significantly from its implementation. Due to the growing popularity of software development in this innovative age, the need for a number of methodologies to adequately manage such complex projects has emerged. One of the most popular choices in these industries is Scrum Framework. The Scrum framework represents the way a project is executed with certain role artifacts and activities that ultimately result in great value for the project. The purpose of this paper is to examine the challenges that arise when using the Scrum framework for software project development. Therefore, in the theoretical part, the concept of agile project management is explained, as well as its methodologies and frameworks, and the Scrum framework is explained in more detail. Also, the specifics of software development project management are clarified, more precisely its trends, determinants and challenges. The empirical part of the paper presents research on the challenges of software development project management using the Scrum framework for agile project management in Croatian IT and ICT companies. The research was conducted with a survey questionnaire examining 39 main challenges with Scrum. It was sent to the mentioned companies and received at least one response (per company) from different relevant persons within the project team. The results of the research showed that, despite the many advantages of the Scrum framework, companies still face a significant number of challenges across the spectrum. Despite this, organizations still choose the Scrum framework and deliver successful software.

Keywords: project management, Scrum framework, IT and ICT industry, software, challenges

SADRŽAJ

1. UVOD

Tijekom posljednja dva desetljeća, obilježena kao doba najvećih inovacija, upravljanje projektima razvoja softvera postaje jedna od najbitnijih komponentni informacijsko-tehnološke (IT) kao i informacijsko-komunikacijsko tehnološke (ICT) industrije. Zbog svog specifičnog načina na koji se organiziraju i primjenjuju znanja, alati i vještine, upravljanje projektima se pokazalo kao odgovarajuće rješenje za probleme koje donosi spomenuto doba inovacije. Ono za cilj ima proizvesti krajnji proizvod u ograničenom vremenskom periodu kroz pokretanje, adekvatno planiranje te kontrolu velikog broja zadataka. Opseg, značaj te složenost nekog projekta glavni su čimbenici zbog kojih se pojedinci odlučuju na upravljanje projektima. Ovisno o pojedinostima tih čimbenika može se odabrati jedno od pristupa upravljanju projektima – tradicionalni, agilni ili pak hibridni, koji se definira kao kombinacija tradicionalnog i agilnog. Projekti unutar IT i ICT industrije često su okarakterizirani kao složeni, neizvjesni i promjenjivi zbog čega se njima ne može upravljati na tradicionalni način. Kratki i iterativni razvojni ciklusi te dobro podnošenje nestabilnih i promjenjivih zahtjeva kupaca pokazalo je agilan pristup kao daleko učinkovitiji kada je riječ o upravljanju projektima u domeni softvera. Jedan od najpoznatijih i najkorištenijih okvira za agilno upravljanje projektima jest Scrum okvir. Scrum okvir realizira projekt kroz uspostavljanje specifičnih uloga, aktivnosti i artefakta koji provođeni adekvatno donose maksimalnu vrijednost konačnom proizvodu koja se najviše očituje u obliku veće kvalitete softvera i manjih troškova ukupnog procesa. Iako je zasnovan na fleksibilnosti i produktivnosti, Scrum okvir također povremeno rezultira negativnim učincima na projekte. Osim što je sam izazov i raditi u stalno promjenjivom IT i ICT okruženju, pokazalo se kako brojni izazovi prilikom korištenja Scrum okvira proizlaze iz samog njegovog procesa i nedovoljno osposobljenog projektnog tima.

1.1. Predmet i cilj rada

Zbog sve veće zamjetne popularnosti pa shodno tome i primjenjivosti Scrum okvira u poduzećima informacijske tehnologije i poduzećima za informacijsku i komunikacijsku tehnologiju, predmet ovoga rada bit će fokusiran baš na tu tematiku. Odnosno, cilj ovog rada je ispitati izazove s kojima se u Republici Hrvatskoj susreću poduzeća prilikom provedbe projekata razvoja novih softvera koristeći Scrum okvir kao jedan od najpoznatijih i najčešće korištenih pristupa za agilno upravljanje projektima u ovom području. Stoga je cilj teorijskog

dijela rada, uz pomoć znanstvene i stručne literature, dati detaljno pojašnjenje agilnog upravljanje projektima i Scrum okvira, te njegovu ulogu u softverskim projektima i nastavno na to prikazati ključna obilježja upravljanja projektima za razvoj softvera. A kako Scrum također povremeno rezultira negativnim učincima za projekte, cilj empirijskog dijela ovoga rada nastoji prikazati ključne izazove s kojima se susreću pojedinci prilikom korištenja Scrum okvira unutar IT i ICT poduzeća.

1.2. Izvori i metode prikupljanja podataka

U istraživanju su razrađeni teorijski i empirijski dio. Teorijski dio temelji se na raznoj znanstvenoj i stručnoj literaturi, te su za potreba rada korišteni znanstveni članci, knjige kao i relevantni internetski izvori. Empirijski dio, osim što se zasniva na podacima iz sekundarnih izvora, također se i sačinjava od podataka prikupljenih putem ankete, upućenim relevantnim osobama u IT i ICT poduzećima. U ovom slučaju su dakle, podaci iz sekundarnih izvora i podaci prikupljeni od strane autorice temelj empirijskog istraživanja. Metode koje su se koristile prilikom sekundarnog istraživanja jesu metoda anketiranja, metoda indukcije i dedukcije, metoda analize i sinteze, metoda kompilacije te metoda komparacije. Metoda anketiranja predstavlja postupak temeljen na anketi iz koje se istražuju i prikupljaju mjerodavni podaci, informacije kao i stavovi i mišljenja ispitanika o predmetu istraživanja. Metoda indukcije se upotrebljava uslijed izvođenja zaključaka na temelju pojedinačnih činjenica i spoznaja, dok se metoda dedukcije koristi kako bi se iz pojedinih općih sudova tj. općih logičkih svojstva između pojmova, otkrile ili dokazale nove činjenice ili zakonitosti. Metoda analize se koristila za objašnjavanje stvarnosti putem raščlanjivanja složenih misaonih tvorevina na jednostavnije sastavne dijelove, te izučavanje svakog dijela za sebe i u odnosu na druge dijelove. Metoda sinteze je postupak znanstvenog istraživanja koja je korištena prilikom objašnjavanja stvarnosti putem spajanja jednostavnih misaonih tvorevina u složene, povezujući izdvojene karakteristike, procese i odnose u jedinstvenu cjelinu u kojoj su njezini dijelovi uzajamno povezani. Metoda kompilacije koristi se radi preuzimanja tuđih rezultata znanstveno-istraživačkog rada, dok je metoda komparacije korištena kako bi bile uspoređene iste ili srodne činjenice, pojave, procesi i odnosi s ciljem dolaženja do zaključaka koji obogaćuju spoznaju.

1.3. Sadržaj i struktura rada

Rad „ Izazovi upravljanja projektima razvoja softvera primjenom Scrum okvira za agilno upravljanje projektima“ podijeljen je u pet osnovnih dijelova.

U prvom dijelu, tj. uvodu, objašnjeni su predmet i cilj rada kao i izvori i metode prikupljanja podataka upotrebljenih za pisanje rada.

Drugi dio rada, obuhvaća teorijski pregled agilnog upravljanja projektima. Predstavljena je povijest agilnog upravljanja projektima kao i njegove glavne značajke. Nadalje, definirane su i opisane različite metodologije i okviri agilnog upravljanja projektima te je naglasak stavljen na Scrum okvir zbog njegove važnosti unutar ovoga rada.

Treće poglavlje opisuje specifičnosti upravljanja projektima razvoja softvera. Konkretnije, tu su izneseni glavni trendovi, kao i temeljne odrednice i u konačnici izazovi koji utječu na upravljanje projektima razvoja softvera.

Empirijsko istraživanje izazova upravljanja projektima razvoja softvera primjenom Scrum okvira za agilno upravljanje projektima započinje u četvrtom poglavlju rada. Poglavlje objedinjuje metodu istraživanja, rezultate istraživanja te ograničenja i preporuke koje su prepoznate u sklopu istraživanja. Metoda objašnjava način na koji se provelo istraživanje, dok rezultati obuhvaćaju detaljan opis i analizu ankete provedene u hrvatskim IT i ICT poduzećima. Ograničenja i preporuke upućuju na probleme prilikom istraživanja kao i što nalažu potencijalne prijedloge za buduća istraživanja.

U konačnici, peto poglavlje donosi zaključak, tj. osvrt na teorijski dio ovoga rada kao i na provedeno empirijsko istraživanje u IT i ICT poduzećima i pruža zaključak o svemu iznesenom.

Poslije zaključka slijedi popis izvora, slika, tablica te životopis autorice.

2. TEORIJSKI PREGLED AGILNOG UPRAVLJANJA PROJEKTIMA

Dobro je poznato kako se današnje poslovno okruženje sve više mijenja i to u svom svakom aspektu. Na prijelazu stoljeća, svijet tehnologije postajao je sve više preplavljen zahtjevima za novim značajkama (Ashmore, Runyan, 2014). To je podrazumijevalo da su poslovni procesi postajali vrlo složeni i pratile su ih dinamične prilike. Takvim okolnostima tradicionalno su se bavili projektni stručnjaci koji bi pokušavali unaprijed prepoznati promjene i shodno njima odrediti sve moguće detalje kako bi njihove organizacije i projekti bili konkurentni na tržištu. Prema tom kontekstu, projektnim stručnjacima postajalo je sve teže na tradicionalan način prilagoditi se brzim promjenama tehnologije i zahtjevima kupaca koji su često postajali nejasni, nepoznati i zahtjevni. Rješenje je došlo u obliku agilnog upravljanja projektima, što je značilo da će razvoj softvera postati više ponavljajući proces (Ashmore, Runyan, 2014).

2.1. Pojmovno određenje agilnog upravljanja projektima

U srži termina agilno upravljanje projektima nalazi se riječ *„agile“*, koja se prema engleskom definira kao „pokretljivost, okretnost“. Izuzev toga, prema latinskom *„agere“* se izvlači značenje „raditi, djelovati“. Povezujući ta dva izvora, dobiva se agilnost koji označava sposobnost brzog pomicanja nečega naprijed. Prema Schwalbe (2016.), agilnost danas znači koristiti metodu koja se temelji na iterativnom i inkrementalnom razvoju, u kojoj se zahtjevi i rješenja razvijaju kroz suradnju. Druga definicija pak tvrdi da je agilno upravljanje projektima skup metoda i metodologija koje pomažu timu da razmišlja više učinkovito, radi učinkovitije i donosi bolje odluke (Stellman, Greene, 2014.). Agilni pristup se može koristiti za razvoj softvera ili u bilo kojem okruženju u kojem su zahtjevi nepoznati ili se brzo mijenjaju. U svojoj osnovi, agilno upravljanje projektima odnosi se na upravljanje utjecajem složenosti i neizvjesnosti na projekt, prepoznavanjem potreba za drastično kraćim vremenskim okvirom između planiranja i izvršenja, kao i da planiranje akcije ne pruža sve detalje njezine provedbe, ali i da su kreativnost i učenje neophodni da bi se dobio smisao okoline (Dybå, Dingsøyr, i Moe, 2014.). Međutim, jednostavnim donošenjem jednokratne odluke za postajanjem „agilnim“ može biti nedovoljno za organizaciju, ili možda čak i za život određenog projekta. Umjesto toga, mora postojati spremnost za promjenu i prilagodbu kao i spoznaja kako najbolje upravljati datom projektnom situacijom unutar određenog okruženja i kulture (Fernandez, Fernandez, 2008.).

2.1.1. Kratka povijest agilnog upravljanja projektima

U poslovnom svijetu pojam *agilni* prvi se put primijenio na projekte razvoja softvera (Schwalbe, 2016.). Agilne metode nastale su ustvari kao reakcija na tradicionalne načine razvijanja softvera. Sve je započelo s takozvanom „krizom razvoja aplikacija“ početkom 1990-ih. Tada je postojalo značajno zaostajanje između poslovne potrebe za aplikacijom i stvarne isporuke softvera. Često se do trenutka objavljivanja konačnog proizvoda tehnologija već razlikovala ili su se zahtjevi kupaca drastično promijenili. To je rezultiralo mnogim neuspjelim projektima i velikim troškovima.

Skupina od 17 vođa softverskih misli počela se neformalno sastajati i razgovarati o načinima za jednostavniji razvoj softvera, bez procesa i dokumentacije kojima su bili opterećeni „slap“ i druge popularne tehnike softverskog inženjerstva tog vremena. Te frustracije oko, naizgled neproduktivnih aktivnosti razvoja softvera, dovele su do danas poznatog agilnog upravljanja projektima. Dakle, početkom 2001. godine, povijest agilnosti došla je u fokus kada se ista grupa, među kojima su Martin Fowler, Jim Highsmith, Jon Kern, Jeff Sutherland, Ken Schwaber i Bob Martin, ponovno sastala, ovaj put na skijalištu u Snowbirdu u državi Utah (www.planview.com). Prema Vanzant Stern (2017.), mnogi su sudionici agilnog saveza imali različite ideje o tome što je činilo agilnu teoriju. Međutim, u roku od samo tri dana grupa je izradila „Manifest za agilni razvoj softvera“ (poznatiji kao *Agile Manifesto*).
Ovaj je manifest izložio četiri ključne vrijednosti:
1. Pojedinci i interakcije nad procesima i alatima
2. Rad softvera preko sveobuhvatne dokumentacije
3. Suradnja kupaca tijekom pregovora o ugovoru
4. Reagiranje na promjenu prateći plan

Dok je agilno upravljanje projektima „poletjelo“ u ranim 2000-ima, viđeno je kako Manifest za agilni razvoj softvera podiže novu „paru“ u 2010-ima. U to je vrijeme povijest agilnog upravljanja projektima bila uobičajena priča među razvojnim timovima, ali između 2012. i 2015. godine mjerni podaci o stvarnom uspjehu počeli su pratiti tu priču. To ne čudi da je tijekom ovog trogodišnjeg razdoblja također uočeno kako agilni način rada premašuje granicu od 50 % u usvajanju, uistinu vodeći razvojni svijet. Agilne metode inovacija revolucionirale su informacijsku tehnologiju. Tijekom posljednjih 25 do 30 godina uvelike su se povećale stope uspjeha u razvoju softvera, poboljšala kvaliteta i brzina izlaska na tržište te povećala motivacija

i produktivnost IT timova (Rigby, Sutherland, Takeuchi, 2016.). Iako budućnost nije poznata, sa sigurnošću se može reći da povijest agilnog način upravljanja projektima još nije dovršena.

2.1.2. Glavne značajke agilnog upravljanja projektima

Sve više i više firmi u različitim industrijama primjećuju glavne značajke, koje se ujedno prepoznaju kao i prednosti, agilnog upravljanja projektima naspram tradicionalnog upravljanja projektima. Ono pruža brojne prednosti organizacijama, projektnim timovima i proizvodima. Te glavne značajke mogu se sažeti u pet skupina.

1. Bolja kvaliteta proizvoda − U Agilnom upravljanju projektima testiranje je integrirani dio faze izvršenja projekta, što znači da je ukupna kvaliteta konačnog proizvoda veća. To se može smatrati najvećom prednošću agilnog pristupa: moguće je koristiti neki dio razvijenog softvera već na kraju prve iteracije, a svaka uzastopna iteracija donosi novu softversku funkcionalnost i time novu vrijednost. To dovodi do zaključka da više ponavljanja donosi više koristi za klijenta, što ih potiče da produže projekt (Stare, 2013.). Uz to, uključivanje kontinuirane integracije i svakodnevnog testiranja u razvojni proces omogućava razvojnom timu da riješi probleme dok su još svježi (Fustik, 2017.). Svakih nekoliko tjedana priče se razvijaju, implementiraju, testiraju i dokumentiraju (Highsmith, 2009.).

2. Uključenost klijenta – Za agilne projekte, presudno je imati stalni pristup korisnicima. Na početku je njihova uloga pružiti i razjasniti zahtjeve, no ubrzo postaje testiranje rješenja i pružanje stalnih povratnih informacija (Robson, 2013.). Prema Fustiku (2017.), za napredak u nekim projektima pri razvoju softvera, treba postojati visoka vidljivost i fleksibilnost za promjene. To podrazumijeva uključenost kupaca u razvojni proces i zadovoljenje potrebe za pokretanjem projekta i dovršenjem projektne misije. Također, projektni tim kupcima demonstrira radne funkcionalnosti u svakom sprintu. Držeći kupca u tijeku i unoseći promjene u skladu s njihovim povratnim informacijama, kupcu se osigurava da konačni proizvod zaista odgovara njegovim zahtjevima.

3. Fleksibilnost − Kada se agilnost uistinu implementira u projektni tim, osnažuje ih neusporedivom fleksibilnošću. Projekti se mogu usmjeriti prema poslovnim ciljevima veće vrijednosti jer je promjene lako ugraditi na kraju svake iteracije, ukoliko se za to iskaže potreba

(Highsmith, 2013.). Ova neusporediva fleksibilnost jedan je od glavnih razloga zašto dinamične organizacije radije koriste agilne metodologije u svom projektu.

4. Poboljšani moral tima − Biti dio samoupravnog tima omogućava ljudima da budu kreativni, inovativni i prepoznati po svojoj stručnosti. Članovi tima zapravo sami odabiru zadatke na kojima žele raditi tijekom iteracije (Ashmore, Runyan, 2014.). To im pruža veću autonomiju kao i zadovoljstvo prilikom rada. Timovi koji se samoorganiziraju moraju imati zajednički fokus, međusobno povjerenje, poštovanje i sposobnost brzog i čestog organiziranja susreta s novim izazovima (Cockburn, Highsmith, 2001.).

5. Smanjen rizik − Jedna od najcjenjenijih značajka agilne metodologije je smanjenje rizika prilagođavanjem korisnikovim potrebama i preferencijama kroz razvojni proces. Agilni tim obično koristi komentare korisnika i iskustvo s poslovno usmjerenim kriterijima za definiranje značajki proizvoda. Nove promjene koje su potrebne mogu se provesti uz vrlo male troškove zbog učestalosti novih koraka koji se proizvode (Fustik, 2017.). Zaključno, može se tvrditi kako stalne aktivnosti u projektu gotovo eliminiraju šanse za apsolutni neuspjeh projekta.

2.2. Metodologije i okviri agilnog upravljanja projektima

Agilne razvojne metodologije postaju sve popularnije zbog svoje usredotočenosti na rukovođenje „*time to market* ograničenjima“ i na sposobnost prilagodbe promjenama tijekom životnog ciklusa razvoja softvera. Dakako, takve metodologije treba prilagoditi kako bi odgovarale potrebama različitih konteksta (Cao, Mohan,Xu, i Ramesh, 2009.). Nastavno na to, slijedi detaljniji prikaz nekih od najčešće korištenih metodologija i okvira upravljanja projektima. Dakle, bit će opisani XP (Extreme Programming), Crystal Metode, Feature-Driven development (FDD) i Dynamic systems development method (DSDM), a s obzirom na to da se cijeli rad temelji na Scrum okviru, on će biti prikazan u sljedećem poglavlju. Također, ne smiju se s njima pomiješati dobro poznati termini Lean pristup i Kanban metoda. To su metode upravljanja projektima koje se ne mogu svrstati u agilne metode upravljanja projektima, ali su s istima usko povezane. Njihov odnos razmatra se na način da su agilne metode i Kanban metoda, potomci Leana, tj. da je Lean nadskup koji dijeli obilježja s agilnim metodama i Kanbanom (PMI 2017.).

2.2.1. XP – Extreme Programming

Stvoreni 1996. godine od strane Kenta Becka uz pomoć Warda Cunninghama i Rona Jeffriesa, principi XP opisani su u Beckovoj knjizi Extreme Programming Explained iz 1999. godine (Layton 2012.). Extreme Programming smatra se jednom od najpopularnijih, a također i najreceptnijih metoda u industriji razvoja softvera čiji je fokus zadovoljstvo kupca. XP timovi postižu visoko zadovoljstvo kupaca razvojem značajki kada ih kupac zatreba. Novi zahtjevi dio su svakodnevice razvojnog tima, a tim je ovlašten nositi se s tim zahtjevima kad god se pojave. U mnogim softverskim okruženjima dinamična promjena zahtjeva jedina je konstanta. Tada će XP uspjeti, dok druge metodologije često neće. Tehnike i principi tj. srž ekstremnog programiranja okarakterizirana je kroz česte i kratke razvojne cikluse, programiranje u paru te redovne izrade i integracijske testove. Osim toga, naglasak je i na kvaliteti, jednostavnosti i izbjegavanju loma koda, na kodiranju samo onoga što je potrebno, te brzim i redovitim povratnim informacijama (Ashmore, Runyan, 2014.). Pored toga, neizostavna je činjenica da je Extreme Programming razvijen uzimajući u obzir pet temeljnih vrijednosti – komunikaciju, jednostavnost, povratne informacije, hrabrost i poštovanje, čiji je cilj da tim programera radi pod zajedničkim načinom razmišljanja kako bi surađivao i stvorio proizvod na visokoj razini (Beck, 2000.). Konačno, primjetno je kako ovu metodologiju, koja veći naglasak stavlja na prilagodljivost nego na predvidljivost, sve veća IT poduzeća sve više usvajaju za kratkoročne projekte kako bi se suočile sa sve promjenjivijim i bržim svijetom.

2.2.2. Crystal Metode

Kristalne metode su obitelj metodologija koje je razvio Alistair Cockburn sredinom 1990-ih. Kolekcija su agilne metodologije za razvoj softvera koje se mogu koristiti za različite projekte ovisno o veličini, složenosti, kritičnosti i broju uključenih ljudi. Ashmore ističe kako se ova metodologija razlikuje od ostalih jer se usredotočuje na ljude umjesto na procese, ali dijeli uvjerenja drugih agilnih metodologija naglašavajući čestu isporuku korisnicima, udruženost timova i fokus na prilagodbu kroz redovita ponavljanja. Svaki član obitelji Crystal označen je bojom koja karakterizira 'težinu' metodologije, tj. što je tamnija boja to je metodologija teža (Amberson i sur., 2002.). Stoga se obitelj metodologija Crystal sastoji od sljedećih varijanti: Crystal Clear, Crystal Yellow, Crystal Orange, Crystal Orange Web, Crystal Red, Crystal Maroon, Crystal Diamond i Crystal Sapphire (blog.scrumstudy.com). Tako redom idu kristalno čiste za male projekte i dalje slijedom za srednje, velike i vrlo velike projekte (Faiza i sur.,

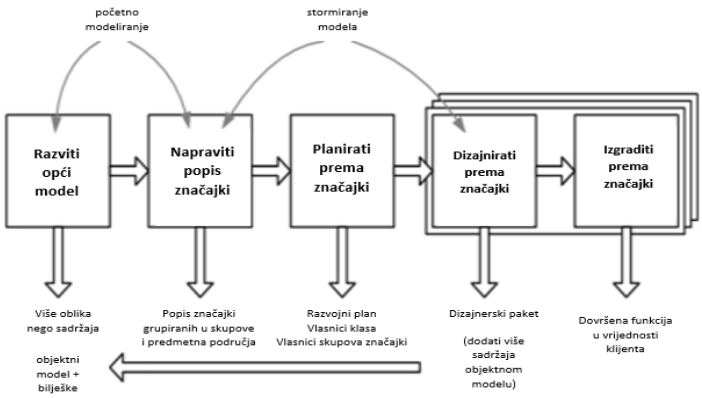
2017.). Boja metode bira se na osnovi mogućih gubitaka prouzročenim padom sustava, dok se razine gubitaka dijele na udobnost (C), diskrecijski novac (D), osnovni novac (E) i život (L). (Amberson i sur., 2002.). Tako se na primjer, metode Crystal Sapphire i Crystal Diamond koriste za velike projekte koji uključuju potencijalni rizik za ljudski život. Postoje, međutim, određene vrijednosti i pravila koja su primjenjiva na cijelu Kristalnu obitelj. Vrijednosti podrazumijevaju visoku toleranciju i usredotočenost na ljude i komunikaciju. S druge strane, pod pravila stoji da razvoj mora biti postepen i trajati kraće od četiri mjeseca te da tim mora provesti sesije razmišljanja prije i nakon projekta (Ashmore, Runyan, 2014). Zbog svoje lakoće i sposobnosti za prilagodbu, Crystal metode su pogodne za korištenje u projektima različitih kategorija.

2.2.3. Feature − Driven development (FDD)

Jeff De Luca i Peter Coad uveli su FDD 1997. godine. Feature Driven Development (FDD) je agilan okvir koji, kako mu samo ime govori, organizira razvoj softvera oko napretka na značajkama. Značajke u kontekstu FDD-a ipak nisu nužno značajke proizvoda u uobičajenom smislu. U takvom procesu promatraju se kao jedinice funkcionalnosti koje se mogu planirati (Hunt, 2006.). S druge pak strane, značajka se interpretira kao bilo koja cijenjena funkcija koju korisnik želi u softveru (Faiza, i sur., 2017.). Proces FDD-a započinje uspostavljanjem ukupnog oblika modela. Nastavlja se serijom dvotjednih iteracija opisanih kao „dizajn po značajki i gradnja po značajki“. Značajke su male, a rezultati "korisni u očima klijenta". Prema PMI (2017.), FDD dizajnira ostatak razvojnog procesa oko isporuke značajki koristeći sljedećih osam praksi:

- Modeliranje objektnih domena
- Razvoj po značajci
- Vlasništvo nad komponentom
- Feature timovi
- Inspekcije
- Upravljanje konfiguracijom
- Redovite gradnje
- Vidljivost napretka i rezultata

Slika 1 FDD životni ciklus procesa



Izvor: izradila autorica rada prema: Shabib i sur., 2017.

FDD životni ciklus procesa vidljiv je na slici 1. Osim što mu se razvojni proces odvija kroz samo pet koraka, također nudi i mogućnosti praćenja i izvještavanja o napretku. Takav agilan okvir za upravljanje projektnima nije baš pogodan za manje projekte, ali s druge strane ima široku primjenu i to ga čini privlačnijim za velike firme i organizacije.

2.2.4. Dynamic systems development method (DSDM)

Dynamic systems development method (DSDM) je agilan okvir za razvoj projekata koji koristi pristup brze primjene u razvoju s velikim naglaskom na kvalitetu. Na taj način pruža ispomoć kroz cijeli životni ciklus razvoja softvera. Poput ostalih agilnih modela DSDM također koristi iterativni i inkrementalni pristup za isporuku kvalitetnog softvera uz stalno sudjelovanje korisnika (Faiza i sur., 2017.). Razvijen je u Ujedinjenom Kraljevstvu 1994. godine od strane praktičara konzorcija koji su pokušavali poboljšati aspekt kvalitete brzih procesa razvoja aplikacija. Svoju prepoznatljivost DSDM očituje u isporuci koja se temelji na ograničenju. Na samom početku projekta, njime će biti određeni vrijeme, kvaliteta i trošak, a potom određivanjem prioriteta definiran opseg projekta kako bi bili zadovoljeni zadani uvjeti (PMI, 2017.). DSDM se temelji na devet ključnih principa koji se prvenstveno vrte oko poslovnih potreba/vrijednosti, aktivnog sudjelovanja korisnika, osnaženih timova, čestih isporuka, integriranog testiranja i suradnje dionika. Zahtjevi su postavljeni na visokoj razini na početku projekta. Prerada je ugrađena u proces, a sve razvojne promjene moraju biti reverzibilne. Zahtjevi se planiraju i isporučuju u kratkim vremenskim okvirima fiksne duljine, koji se nazivaju i iteracije, a zahtjevi za DSDM projekte definiraju prioritete koristeći se sljedećim pravilima: M – „must have requirements“ tj. obvezni zahtjevi; S – „should have if at all possible“ potrebni ukoliko su uopće mogući; C – „Could have but not critical“ mogući ali nisu nužni,; W – „won‘t have this time, but potentially later“ potencijalni, najmanje važni zahtjevi (Fustik, 2017.). Nadalje, potrebno je napomenuti važnost procesa kojim DSDM funkcionira. Naime, DSDM kombinira upravljanje projektima i aktivnostima povezanim s razvojem proizvoda zajedno u jednom procesu. Taj proces, odnosno bolje rečeno životni ciklus DSDM-a sastoji se od šest faza i to su faza prije projekta, studija izvodljivosti, poslovna studija, iteracija funkcionalnog modela, dizajniranje i razvoj iteracija, te provedba i post projektna faza (Faiza i sur., 2017.). Svoju korisnost i primjenjivost iskazuje kroz odgovarajuće smjernice poput kontrole rizika i različitih razvojnih tehnika te je zbog toga idealan, kako za mala tako i za veća poduzeća. Osim toga, važno je kako se DSDM koristi pretežito u projektima za razvoj softvera,

ali ponekad i u svrhe ne IT projekata i zbog toga se često smatra komplementarnim pristupom ekstremnom programiranju.

2.3. Scrum okvir za agilno upravljanje projektima

Vjerojatno najprepoznatljivija riječ kada se agilnost primjenjuje na upravljanje projektima je Scrum. Scrum je agilna metodologija koja se može primijeniti na gotovo bilo koji projekt. Iako se izvorno koristio za razvoj softvera, Scrum teorija sada se koristi u svim vrstama poslovnih pothvata. Prema Harvard Business Reviewu (1986.), prvi spomen Scruma seže još iz 1986. godine. Scrum nije svojevrsni standardizirani postupak prema kojem se metodično slijedi niz uzastopnih koraka za koje se jamči da će se na vrijeme i uz proračun dobiti visokokvalitetni proizvod koji oduševljava kupce. Umjesto toga, Scrum je okvir za organiziranje i upravljanje radom. Temelji se na skupu vrijednosti, principa i prakse koje pružaju temelje za organizacije da dodaju svoje jedinstvene primjene relevantnih i specifičnih inženjerskih praksi čiji će rezultat biti verzija Scruma koja je jedinstvena (Rubin, 2012.).

Scrum okvir mogao bi se sažeti na sljedeći način:

- Vlasnik proizvoda stvara prioritetni popis želja koji se naziva „product backlog“ ili drugim riječima Popis stavki koje proizvod treba imati
- Tijekom planiranja sprinta tim povuče mali dio s vrha popisa želja, tj. Popis stavki za sprint, i odlučuje kako implementirati te dijelove
- Tim ima određeno vrijeme, sprint, da završi svoj posao - obično dva do četiri tjedna - ali sastaju se svaki dan rade procjene napretka
- Kroz cijeli proces, Scrum Master drži tim usredotočenim na svoj cilj
- Na kraju sprinta, rad bi trebao biti potencijalno dostupan, kao i spreman predati kupcu, staviti na policu trgovine ili pokazati dionicima
- Sprint završava pregledom i retrospektivom sprinta
- Kako započinje sljedeći sprint, tim odabire još jedan dio Popisa i ponovno počinje raditi

Ciklus se ponavlja dok se ne završi dovoljno stavki u Popisu ili je proračun iscrpljen ili se približava rok za završetak projekta. Koji od ovih prekretnica označava kraj rada posve ovisi o projektu. Bez obzira koji od navedenih razloga zaustavlja rad, Scrum osigurava da je najvažniji dio posla obavljen kada projekt završi (Schwalbe, 2016.).

Sami Scrum postupci utjelovljeni su u određenim ulogama, aktivnostima i artefaktima. To će biti opisano u nastavku i može se prikazati sljedećom slikom.

Slika 2 Sažetak Scrum uloga, aktivnosti i praksi

Izvor: izradila autorica prema froggyads.com

2.3.1. Scrum uloge

Postoje samo tri Scrum uloge: vlasnik proizvoda, tim i Scrum Master. Sve odgovornosti upravljanja u projektu podijeljene su između ove tri uloge. U nastavku su opisani i pobliže pojašnjeni zadaci i obilježja spomenutih uloga:

a) Vlasnik proizvoda

Uloga i odgovornost vlasnika proizvoda jedna je od najvažnijih prilikom provođenja Scrum projekata, a često i najteža. On je odgovoran za financiranje projekta tijekom njegovog životnog ciklusa i postavljanje zahtjeva i ciljeva projekta. Njegova ključna uloga je maksimizirati učinak tima i učinak svakog zadatka, na temelju povrata na ulaganje (ROI) (Sverrisdottir, Ingason, Jonasson, 2014.). Provođenje spomenutog često se uvelike razlikuje među poduzećima, Scrum timovima i pojedincima. Prema Rubinu, vlasnik proizvoda jedini je ovlašten i odgovoran za odlučivanje o tome koje značajke i funkcionalnosti će se graditi i kojim redoslijedom ih treba graditi. Njegova je zadaća da održava i priopćava svim ostalim sudionicima jasnu viziju onoga

što Scrum tim pokušava postići. Mahalakshmi i Sundararaja (2013.) uloge vlasnika proizvoda definirali su na sljedeći način:

- djeluje kao predstavnik kupca
- surađuje s timom
- održava Popis stavki koje proizvod treba imati
- daje prioritet zahtjevima proizvoda
- izravna je veza između tima i poduzeća
- definira značajke proizvoda
- odlučuje o datumu i sadržaju proizvoda
- odgovoran je za dobit
- prioritizira i prilagađa značajke kako i kada je to potrebno
- odgovoran za konačno odobrenje
- prihvaća ili odbija rezultate rada

Vlasnik proizvoda aktivno surađuje sa Scrum Masterom i developerskim timom i mora biti raspoloživ za pružanje odgovora na pitanja ubrzo nakon što ona budu postavljena (Rubin, 2012.). Kao takav, snosi odgovornost za ukupni uspjeh projekta, a da bi u tome uspio, čitava organizacija mora poštivati njegove odluke. Te su odluke vidljive u sadržaju i redoslijedu Popisa, te kroz provjerljivi prirast na Sprint Reviewu (www.scrum.org).

b) Scrum Master

Scrum Master je osoba koja osigurava produktivnost tima olakšavajući dnevni Scrum, omogućujući blisku suradnju u svim ulogama i funkcijama, te osim toga uklanja prepreke koje sprječavaju tim da bude učinkovit. On ima autoritet nad procesom, ali ne i nad ljudima u timu (Schwable, 2016.) stoga ova uloga nije ista kao i tradicionalna uloga voditelja projekata jer Scrum Master funkcionira kao vođa, a ne kao menadžer (Rubin, 2012.). Scrum Master odgovoran je za poštivanje pravila Scrum okvira u projektu. On osigurava da se ta pravila razumiju unutar organizacije i da se posao obavlja prema njima. Palacio (2021.) tvrdi kako ova uloga savjetuje i pruža potrebnu obuku i vlasniku proizvoda i timu, a također da oni konfiguriraju, dizajniraju i kontinuirano poboljšavaju agilne prakse organizacije. Cilj je da se tim i klijent mogu organizirati i raditi samostalno. Dobri Scrum majstori predani su Scrum temeljima i vrijednostima, ali ostaju fleksibilni i otvoreni za mogućnosti tima, da poboljša svoj tijek rada (www.atlassian.com). Nadalje, odgovornosti Scrum Mastera su moderirati dnevne sastanke za Scrum, upravljati poteškoćama grupne dinamike koje se mogu pojaviti u timu te

rješavati sve prepreke otkrivene tijekom svakog dnevnog Scruma kako bi sprint mogao napredovati. On osigurava primjenu i učinkovit Scrum okvir moderiranjem svakodnevnih sastanaka Scruma i rješavanjem problema koji su tijekom njih identificirani. On pomaže timu kako bi se mogli glatko kretati prema naprijed. Imati posvećenog Scrum Mastera preporučuje se kada vlasnik proizvoda ili tim imaju malo iskustva s korištenjem Scruma ili u velikim organizacijama s konstantnim protokom izmijene ili obuke osoblja (Palacio, 2021.). Biti Scrum majstor ne odgovara svima: nekim ljudima nije ugodna „vidljivost“ povezana s ulogom ili preuzimanjem inicijative koja je neophodna za uspjeh (Schwaber, Beedle 2002.).

c) Razvojni tim

Razvojni tim je višefunkcionalni, samoorganizirajući tim koji se sastoji od članova koji posjeduju sve što im je potrebno za isporuku radnog proizvoda bez ovisnosti o drugima izvan tima (PMI, 2017.). Oni su kros funkcionalni, a to implicira da ubrajaju sve relevantne stručnjake za isporuku proizvoda pri završetku svakog sprinta (Schwaber, Sutherland, 2017.) i reduciraju troškove kašnjenja i one troškove koji nastaju uslijed velikih hijerarhija orijentiranih ka disciplini (Keith, 2010.). Kako bi procijenio i odabrano najbolji način za postizanje ciljeva postavljenih od strane vlasnika proizvoda, razvojni tim se samoorganizira. Nadalje, jedan takav tim obično ima pet do devet ljudi. Njegovi članovi moraju kolektivno imati sve vještine potrebne za stvaranje kvalitetne i radne softverske opreme. Naravno, Scrum se može koristiti za razvojne napore koji zahtijevaju mnogo veće timove. Međutim, umjesto da postoji jedan Scrum tim s, recimo 35 ljudi, vjerojatnije bi bilo organizirati četiri ili više Scrum timova, svaki s devet ili manje ljudi (Rubin, 2012.).

2.3.2. Scrum aktivnosti

Suština Scruma leži u sprintu, a sprint predstavlja vremenski limit iz kojeg se formira upotrebljiv i finaliziran proizvod kojega je potencijalno moguće pustiti u rad. Dakle Scrum postupak se sastoji od pet glavnih aktivnosti, a to su planiranje sprinta, dnevni Scrum, sprint review, retrospektive sprinta i sami sprint (Schwaber, Sutherland, 2017.). Sprint započinje planiranjem sprinta i obuhvaća razvojne aktivnosti tijekom sprint (naziva se sprint izvedba), a završava pregledom i retrospektivom. Dakle, svaki sprint započinje takozvanim sastankom za planiranje sprinta na kojem sudjeluju vlasnik proizvoda, razvojni tim i Scrum Master. U prvom dijelu sastanka događaju se dvije velike aktivnosti. Prvo, skupina definira Popis stavki koje proizvod treba imati. Broj stavki u tom Popisu vjerojatno će biti veći nego što razvojni tim može dovršiti u kratkotrajnom sprintu. Iz tog razloga, na početku svakog sprinta, razvojni tim mora

utvrditi podskup Popisa za koje vjeruje da ih može dovršiti. Kao drugo, da bi se uvjerili da se razvojni tim razumno obvezao, članovi tima stvaraju drugi Popis tijekom planiranja sprinta, tzv. sprint Backlog. Popis stavki za Sprint opisuje, kroz niz detaljnih zadataka, kako tim planira osmisliti, izgraditi, integrirati i testirati odabrani podskup značajki s Popisa stavki koje proizvod treba imati tijekom tog sprinta (Rubin, 2012.). Prema Cervone (2011.), na ovom dijelu sastanka, svaki član tima kratko odgovara na tri pitanja i utvrđuje se što se radilo od zadnjeg Scruma, što će se raditi do sljedećeg Scruma, kao i što sprječava tim da nastavi sa svojim poslom. Po tom, slijedi sprint izvedba, gdje razvojni tim obavlja zadatke potrebne za realizaciju odabranih značajki. Svaki dan tijekom izvođenja sprinta, članovi tima pomažu upravljati tijekom posla sinkronizacijom, pregledom, i aktivnostima adaptivnog planiranja poznatih kao dnevni Scrum. Na kraju izvođenja sprinta tim je proizveo potencijalno isporučiv *product inkrement* koji predstavlja neke, ali ne sve, vizije vlasnika proizvoda. Scrum tim dovršava sprint izvodeći dva pregleda i prilagođavanja aktivnosti. U prvom, nazvanom *sprint review*, dionici i Scrum tim će pregledati proizvod koji se gradi. U drugom, nazvanom retrospektiva sprinta, Scrum tim pregledava Scrum postupak koji se koristio za stvaranje proizvoda. Ishod ovih aktivnosti mogle bi biti prilagodbe koje će ući u Popis stavki za proizvod ili biti uključene u razvojni proces tima. U ovom se trenutku ponavlja Scrum sprint ciklus, koji počinje iznova s razvojnim timom određujući sljedeći najvažniji skup Popisa koji može završiti (Rubin, 2012.). Pri završetku sprinta svaki novonastali inkrement treba biti realiziran, tj. mora biti u funkcionalan. Svaki inkrement je pomak naprijed ka cilju i viziji (Schwaber, Sutherland, 2017.). Nakon što je dovršen odgovarajući broj sprintova, vizija vlasnika proizvoda će se ostvariti i rješenje se može objaviti.

2.3.3. Scrum artefakti

Posljednja značajna determinanta Scrum okvira su Scrum artefakti koji uključuju Popis stavki za proizvod, Popis stavki za Sprint i grafikon spaljivanja (*Burndown chart*). Scrum artefakti su sredstva koja omogućuju transparentnost kroz zajedničko razumijevanje djela. Osim toga, pružaju timovima mogućnost pregleda i prilagodbe (Gonçalves, 2018.). Popis stavki za proizvod je uređeni popis zahtjeva koji se održava za dobrobit proizvoda. Za razliku od tradicionalnih projekata, ovim popisom upravlja vlasnik proizvoda. Dakle, sve stavke koje sadrži naručio je odnosno zahtijevao vlasnik proizvoda. On odlučuje o prioritetima i može promijeniti popis tijekom ili na kraju sprinta. Transparentan je za sve stakeholdere (Mahalakshmi, Sundararajan, 2013.). Ovaj Scrum artefakt definira što je sljedeće najvažnije

izgraditi. Glavna značajka mu je da se stalno ažurira i poboljšava. Drugi pak važan artefakt je Popis stavki za Sprint. Prema Cervone (2011.), to je podskup definiranih stavki Popisa za proizvod koji su objašnjeni kao dio posla za određeni sprint. Popis stavki za sprint je dakle popis radova za razvojni tim koji se moraju obaviti tijekom sljedećeg sprinta. Kao što je već opisano u prethodnom podnaslovu, Popis stavki za proizvod opisuje, kroz niz detaljnih zadataka, kako tim planira osmisliti, izgraditi, integrirati i testirati odabrani podskup značajki Popisa tijekom tog sprinta (Rubin, 2012.). On čini vidljivim sav rad koji razvojni tim identificira kao neophodan za postizanje Sprint cilja (www.visual-paradigm.com). Važna je činjenica kako se jednom izdan zahtjev ne smije mijenjati za vrijeme izvođenja sprinta. Konačno, Burndown chart je grafikon koji, osim što se ažurira svaki dan, prikazuje preostali rad u Popisu stavki za sprint. Također, pruža brze vizualizacije za referencu (Mahalakshmi, Sundararajan, 2013.). Postoje tri vrste toga grafikona. Dakle, Sprint burndown, product burndown i release burndown su grafikoni koji se koriste za označavanje napretka i provjeru izgaranja. Polaze od maksimalnih bodova i s vremenom se pomiču prema nuli (Srivastava, Bhardwaj, Saraswat, 2017.). Ne iznenađuje stoga što se grafikon sagorijevanja proizvoda koristi za označavanje ukupnog napretka projekta.

3. SPECIFIČNOSTI UPRAVLJANJA PROJEKTIMA RAZVOJA SOFTVERA

Kako bi proizvodi brzo, tj. u kratkom roku bili plasirani na tržište, timovi za razvoj softvera oslanjaju se na djelotvorno upravljanje projektima da pojednostave svoje tijekove rada. Upravljanje projektima razvoja softvera disciplina je ocjenjivanja karakteristika softvera, odabir najboljeg životnog ciklusa za razvoj softvera, a zatim odabir odgovarajućeg pristupa upravljanju projektom kako bi se osiguralo zadovoljenje potreba kupaca za isporuku poslovne vrijednosti, što je moguće učinkovitije (Wysocki, 2014.). Upravljanje softverskim projektima posvećeno je planiranju, raspodjeli resursa, izvođenju, praćenju te isporuci softvera i web projekata. Upravljanje projektima razvoja softvera razlikuje se od tradicionalnog upravljanja projektima po tome što softverski projekti imaju jedinstven proces životnog ciklusa koji zahtijeva više rundi testiranja, ažuriranja i povratnih informacija korisniku. Većina projekata povezanih s IT-om upravljana je na agilan način da bi održali korak sa sve većim tempom poslovanja (www.wrike.com). Agilni razvoj softvera predstavlja novi pristup planiranju i upravljanju softverskim projektima. On stavlja manji naglasak na planiranje unaprijed i strogu kontrolu dok se više oslanja na neformalnu suradnju, koordinaciju i učenje (Dybå, Dingsøyr, Moe, 2014.).

3.1. Trendovi koji utječu na upravljanje projektima razvoja softvera

Nove generacije praksi i trendova u upravljanju projektima za razvoj softvera rezultirale su mijenjanjem konvencionalne IT industrije i djelovale na njen brzi razvoj (Akbar, Safdar, 2015.). Radi lakšeg objašnjavanja trendovi su identificirani kroz dvije skupine − trendovi vanjskih čimbenika i trendovi unutarnjih čimbenika. Glavni bi bili globalizacija, sveprisutno računarstvo u oblaku, *outsourcing* te trendovi vezani uz agilne metodologije, testiranje softvera, ispitivače softvera, i ostali na kojima neće biti fokus za ovaj rad ali ih je važno spomenuti poput distribuiranog razvoja softvera, AI, poboljšanje procesa i standardizacija, razvoj mobilnih aplikacija, društveno umrežavanje itd.

3.1.1. Trendovi vanjskih čimbenika

Nedavni trendovi poput globalizacije, *outsourcinga* i računarstva u oblaku kreiraju projektnim menadžerima i njihovim timovima jedinstvene prilike za razvoj u ovom brzo mijenjajućem svijetu. Trendovi kao i njihova obilježja slijede redom:

a) Globalizacija

Jedna od uočljivih karakteristika IT industrije početkom 21. stoljeća je globalizacija. Globalizacija je povećala povezanost i integraciju u političkim, kulturnim, društvenim, ekonomskim i tehnološkim sustavima između brojnih naroda, korporacija, kućanstava i pojedinca. Instantnu interakciju velikog broja ljudi, koji se kreće u milijardama, učinile su mogućim niže političke i trgovinske barijere te digitalna revolucija. Za pojedince i mala poduzeća to znači kako napokon mogu konkurirati velikim korporacijama. Osim toga, Friedman raspravlja o trendu rasta „učitavanja sadržaja“ na Internetu uz kojeg ljudi mogu razmjenjivati znanja putem internetskih blogova, podcasta i softvera otvorenog koda (Schwalbe, 2016). Primarna značajka globalizacije u domeni razvoja softvera je manji trošak rada. Tako je na primjer cijena rada po glavi stanovnika u većini azijskih zemalja izrazito niska, dok su troškovi razvoja softvera u SAD -u, Kanadi i većini europskih zemalja znatno veći nego u većini azijskih zemalja (Batra, Sin, Tseng, 2006.). Druga karakteristika ovoga trenda je mogućnost zapošljavanja vještih i dobro obrazovanih programera iz Indije ili Kine koje su se pokazale kao zemlje sa puno visoko kvalificiranih programera. Treći čimbenik je sposobnost stvaranja virtualne korporacije i virtualnih timova te u kratkom vremenu iskoristiti tržišne mogućnosti pogodne za projekte u domeni razvoja softvera (Herbsleb, Moitra, 2001.). Osvrćući se na trenutnu situaciju s COVID-19 virusom koja je dovela do učinka virtualne prilagodbe (Klonek i sur., 2021.), projekti se unatoč okolnostima, nisu zaustavljali i može se reći kako su se virtualni timovi pokazali kao dobro rješenje za upravljanje projektima za razvoj softvera. Četvrta važna odrednica ovoga trenda je sposobnost korištenja razlike u vremenskim zonama za postizanje „*follow the sun*“ razvoja. To može osigurati doslovce razvoj 24 sata na dan, koji može dovesti do poboljšanja izvedbe. Međutim, zadatke u različitim fazama životnog ciklusa softvera treba pažljivo i dobro podijeliti (Cho, 2007.). U konačnici, globalizacija ima veliki utjecaj na industriju razvoja softvera i zbog brojnih sadašnjih i nadolazećih trendova drastično mijenja način na koji se upravlja projektima u toj domeni.

b) Outsourcing

Outsourcing je važan trend u današnjem poslovnom i IT okruženju prvenstveno zbog privlačne ideje o potencijalnom smanjenju troškova prilikom izvođenja projekata za razvoj softvera (Erickson, Ranganathan, 2006.). *Outsourcing* nerijetko asocira na vanjsko ugovaranje te ostale poslovne odnose s partnerskim firmama, recimo dobavljačima. Podrobnija pak interpretacija otkriva kako *outsourcingom* pojedini poslovni procesi bivaju dodijeljeni specijaliziranim partnerskim poduzećima čija je glavna karakteristika mogućnost obavljanja tih poslovnih procesa po većoj kvaliteti i jeftinije, uz očuvanje komunikacije s organizacijom s kojom je potpisan ugovor o *outsourcingu*. Takvim se načinom poduzeće, koje se odlučilo za izdvajanje određenih poslovnih procesa, može usmjeriti ka ključnim poslovnim procesima (Liović, 2016). Jasno, vanjski *outsourcing* značajno komplicira projekte za razvoj softvera i povećava potrebu za snažnim projektnim sposobnostima upravljanja zbog proširenih komunikacijskih linija, potencijalnih kulturalnih nesporazuma i rizik od loše iskomuniciranih ciljeva (Rao, 2004.). Također, zbog sve češće upotrebe *outsourcinga* u projektima, primarni je zadatak projektnih menadžera da budu što bolje upućeni o brojnim globalnim pitanjima, što obuhvaća rad kao i upravljanje i vođenje virtualnih timova.

c) *Cloud computing* – računarstvo u oblaku

Kako bi se olakšao rad na daljinu, firme su morale implementirati odgovarajuću infrastrukturu i podržati zaposlenike u stvaranju radnog okruženja u svojim matičnim uredima. Kao rezultat toga, nastao je trend tj. potražnja za uslugama i alatima u oblaku. Poduzeća za razvoj softvera aktivnije koristiti SaaS, IaaS i PaaS rješenja za izradu aplikacija, upravljanje timovima i komunikaciju jer se te usluge mogu lako usvojiti i uvesti (www.sam-solutions.com). S druge strane, računalstvo u oblaku nije samo tehnološki pojam koji se odnosi na podatke, obradu ili iskustva koja „žive“ negdje u internetu, nego i tiha revolucija u načinu na koji firme rade s podacima i aplikacijama u procesima pronalaska, razvoja, implementacije, skaliranja, ažuriranja, održavanja i plaćanja resursa koji su podvrgnuti promjenama (Yung, Lin, 2015.). Dobro je dokumentirano kako se primjerice stvaranje vlastitih podatkovnih centara teško može mjeriti po pitanju cijene kakvu mogu ponuditi pružatelji računalstva u oblaku. Pružanje računalnih resursa onima kojima je to najpotrebnije u vrijeme kada im je potrebno primarni je rezultat učinkovitosti računalstva u oblaku (Maximilien, Campos, 2012.).

Słoniec (2015.), glavne razloge za pojavljivanje trenda računalstva u oblaku za upravljanje softverskim projektima objašnjava na sljedeći način:

- Nema potrebe za posebnim poslužiteljem za upravljanje projektima
- Sigurno skladištenje podataka koji se odnose na projekt
- Sposobnost korištenja najnovije verzije aplikacije koja podržava upravljanje projektima
- Bez posebne licence za softver
- Pristup projektu s bilo koje lokacije koja ima pristup internetu
- Automatizacija određenih procesa
- Pristup velikoj računalnoj snazi neophodnoj za provedbu velikih i složenih projekata
- Sposobnost daljinskog rada članova projektnog tima ako je potrebno
- Tehnička pomoć pružatelja usluga u oblaku

Popularnost računarstva u oblaku nevjerojatnom brzinom je promijenila i preoblikovala mehanizme i vrijednosti poslovanja firmi (Schneckenberg i sur., 2021.). Ovaj trend pokazao se neizbježnim. Industrije diljem svijeta primjenjuju najnovije tehnološke inovacije kako bi pomogle svom poduzeću ostvariti što veći uspjeh. Budući da ova rješenja pružaju veću fleksibilnost i bolje mogućnosti upravljanja podacima, poduzeća sada shvaćaju značaj računalstva u oblaku za upravljanje projektima razvoja softvera.

3.1.2. Trendovi unutarnjih čimbenika

Nadalje identificirana su tri glavna trenda koja su se pojavila u direktnom doticaju sa razvojem softvera. Navedeni su i objašnjeni naniže:

a) Agilne metodologije

S obzirom da je ova tematika obrađena unutar prvog poglavlja neće se ovdje dalje razrađivati ali će se istaknuti njezina važnost unutar industrije za razvoj softvera. Nastavno na to, valja se reflektirati na činjenicu kako je razvoj softvera brži nego ikada do sada što je učinilo tržište preplavljeno nebrojivim softverskim aplikacijama. U takvim okolnosti, tradicionalni pristupi razvoja softvera nisu u stanju ispuniti zahtjeve najnovijih trendova u IT industriji (Akbar, Safdar, 2015.). Agilne metode u domeni razvoja softvera postaju sve popularnije po pitanju životnog ciklusa. Veliki trend pokazao se odbacivanje prakse "jedna veličina odgovara svima" ranijih metoda, poput SSADM verzija tri i četiri. Umjesto toga, nastaje pomak na metode koje obuhvaćaju više „alternativnih modela životnih ciklusa“ (Redmond-pyle, 1996.). Relevantno je kako je najvažnija značajka agilne metode brža isporuka softvera kao i mogućnost značajnog poboljšanja kvalitete razvoja i produktivnosti (Talby i sur., 2006.). Agilni okvir Scrum i njegov

uspjeh u razvoju složenih hardverskih i softverskih proizvoda u različitim industrijama čini ga uvjerljivim okvirom za usvajanje u brojnim organizacijama.

b) Mijenjanje načina testiranje softvera

Drugo, testiranje softvera postaje integralna aktivnost u procesu razvoja softvera. Testiranje softvera je neophodno u osiguravanju kvalitete softvera (Cohen i sur., 2004.). Tradicionalno se na testiranje gledalo kao na različitu i odvojenu fazu na kraju procesa razvoja softvera. Međutim, evolucija razvoja metodologija softvera ukazuju da testiranje više nije kulminirajuća aktivnost razvojnog procesa. Pyhäjärvi i Rautiainen (2004.) tvrde da je testiranje „integralna aktivnost u razvoju softvera“ i preporučuju „testiranje treba uključiti u razvoj softvera”. Schach (1996.) također predlaže da „treba provesti testiranje tijekom životnog ciklusa softvera ”.

c) Veća važnost ispitivača softvera

Prema Zhang, Hu, Dai i Li-u (2020.) treći trend se odnosi na testere tj. ispitivače softvera koji sada igraju važniju ulogu u razvoju softvera kao izravni rezultat trenda opisanog iznad. Tvrde kako se dosadašnja uloga testera proširila na dva načina. Prvo, odgovornosti ispitivača napredovale su iz „pronalaženje pogrešaka“ (Myers, 1979.) do „osiguranja kvalitete“ (Hetzel, 1984.) pa do „provjere i validacije koda“ (Bentley, 2005.). Drugo, testeri su angažirani ranije i tijekom cijelog procesa razvoja softvera, što se pokazalo korisnim za performanse projektnog tima. Na primjer, Zhang, Hu, Dai i Li (2020.) pronalaze kvantitativne dokaze koji su započeli pokretanje testiranja ranije u razvojnom procesu i tada je izvedba projekta, u smislu troškova i vremenskog ciklusa, poboljšana bez žrtvovanja ukupne kvalitete softvera. Ranije uključivanje ispitivača također omogućuje timu da prije uoči nedostatke u procesu upravljanja razvojnim softverskim projektima. To pomaže u smanjenju troškova razvoja jer, pokazalo se kako se otklanjanjem nedostataka otkrivenih u kasnijim fazama povećavaju troškovi za jedan ili više redova veličina (Tyran, 2006).

3.2. Odrednice upravljanja projektima razvoja softvera

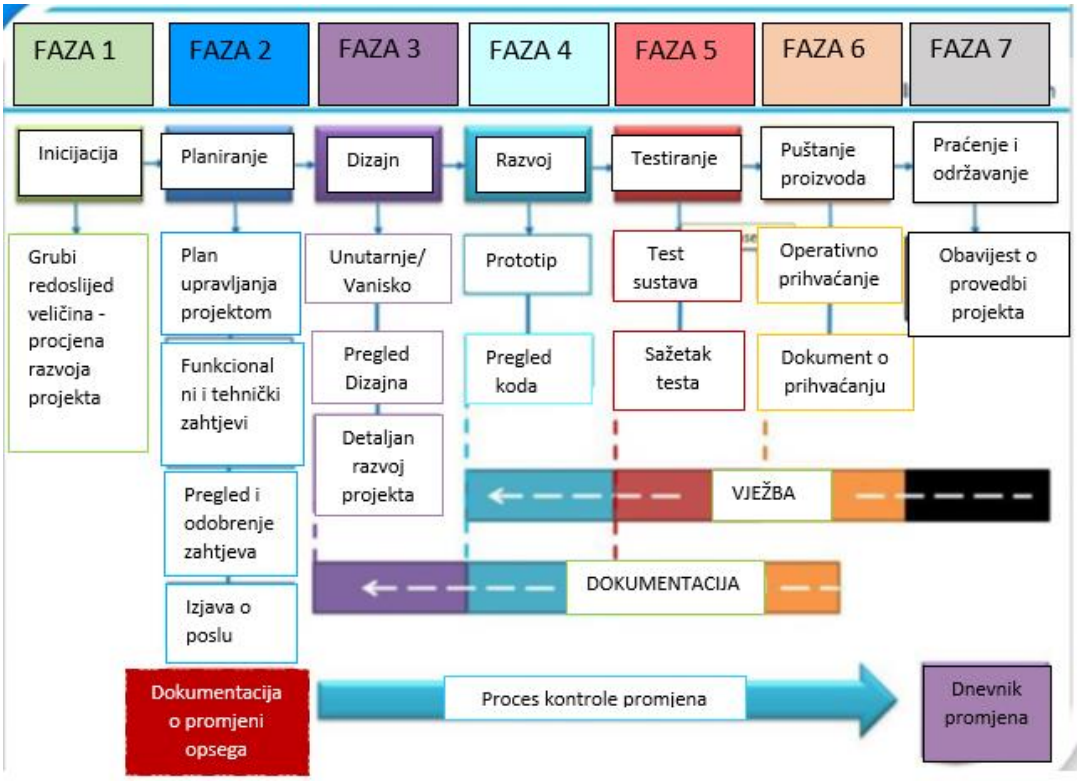
Nijedna druga upravljačka aktivnost ne može imati više koristi od učinkovitog upravljanja projektima nego razvoj softvera. Praktički svi napori poduzeti za razvoj softvera u toj kategoriji smatraju se projektima (Jurison, 1999.). No, upravljanje projektima razvoja softvera susreće se sa više izazova i poteškoća od tradicionalnih projekata. Upitno je zašto je samo 32 % softverskih

projekata bilo uspješno u 2009. godini. Preostali projekti su ili propali ili su izazvali prekoračenje troškova, prekoračenje rasporeda i nedostatak kvalitete. To znači da postoji nešto posebno u softverskim projektima u usporedbi s tradicionalnim projektima. Ta posebnost vidi se u tome da su softverski projekti nevidljivi, složeniji, zahtijevaju stalnu nadogradnju, tehnološki ovisniji i teško ih je procijeniti. Softverske projekte trebaju provoditi ljudska bića. Automatsko generiranje koda moguće je do određene mjere. Međutim, prednosti softverskih projekata su fleksibilne, ponovna upotreba komponenti je veća, skalabilna, jednostavna za izradu sigurnosnih kopija i imaju mogućnost proširivanja koda, na temelju programskog jezika kojim se koriste. Ovih dana postoje čak i programski jezici i okviri neovisni o platformi, iako su se neke od softverskih organizacija poput Microsofta, IBM –a, SUN i ORACLE Corporation zalagale za upravljanje projektima softvera među kategorijama proizvoda. Organizacije poput Accenture, Infosys i Tata Consultancy Services (TCS) sazrijele su u sektoru softverskih usluga (Sudhakar, 2010).

3.2.1. Etape razvoja softverskog projekta

Svaki softverski projekt kreće se i napreduje kroz različite faze prije nego što se zatvori. Životni ciklus softvera, stoga predstavlja proces razvoja softvera koji softverska industrija koristi za projektiranje, razvoj i testiranje visokokvalitetnih softvera. Ono uključuje sljedeće faze - pokretanje projekta, planiranje projekta (analiza i skupljanje zahtjeva), dizajn, razvoj, testiranje, puštanje proizvoda te praćenje i kontrola (održavanje) projekta. Ove faze projekta mogu se vidjeti na slici 3.

Slika 3 Životni ciklus razvoja softvera



Izvor: izradila autorica rada prema www.pdfprof.com

Prva faza zove se faza pokretanje i tijekom nje se priprema povelja projekta. Povelju projekta može pripremiti projektni sponzor ili voditelj projekta. U vrijeme pripreme povelje projekta sponzor identificira voditelja projekta i dodjeljuje mu projekt. Povelja projekta sastoji se od naziva projekta, opisa, izvedivosti, definiranja voditelja projekta, identifikacija nekih od dionika, pretpostavki, ograničenja, troškova i okvirnog vremena projekta (Sudhakar, 2010). Također, provodi se i analiza elemenata softvera i specificiranje zahtjeva što predstavlja aktivnost identificiranja problema koje je potrebno razriješiti novim softverom (Čubranić i sur., 2013.). Izdavanje povelje projekta označava početka istog. U osnovi, ovu fazu sačinjavaju četiri glavna zadatka – priprema povelje projekta, identifikacija i dokumentiranje problematike i potreba projekta, odabir voditelj projekta te utvrđivanje organizacijske politike relevantne za izvođenje softverskog projekta (Sudhakar, 2010.).

Druga procesna etapa, planiranje (prikupljanje zahtjeva), određuje kako će se projekt kretati naprijed nakon što su izvedivost projekta, opis i povelja dovršeni. Smisao planiranja u upravljanju softverskim projektima je komunicirati točno ono što će se raditi u projektu (Luckey, Phillips, 2006.). Tijekom ove faze, voditelj projekta priprema plan projekta te dijeli projekt na dijelove koristeći strukturu raščlanjivanja poslova (WBS). Nadalje, u ovoj se fazi

identificiraju aktivnosti, kompletan raspored projekta te proračun troškova za projekt. Voditelj projekta identificira rizike, vrši kvalitativnu, kvantitativnu analizu rizika i priprema plan odgovora na rizik (Sudhakar, 2010). Ova faza najčešće kreće s odabirom jedne od sljedećih formalnih metodologija za upravljanje softverskim projektima: Rolling wave planiranje, Scrum ili XP koja će dalje određivati tijek projekta.

U fazi analize raščlanjuju se rezultati u povelji projekta na visokoj razini na detaljnije poslovne zahtjeve. Faza analize također je dio projekta gdje se identificira opći smjer kojim će se projekt kretati kroz izradu dokumenata strategije projekta. U mnogim slučajevima prikupljanje zahtjeva i analiza mogu se provesti istovremeno (www.learntek.org).

U fazi dizajna, specifikacije softverskih zahtjeva (SRS) sada su transformirane u plan projektiranja sustava, koji sadrži detaljan i cjelovit skup specifikacija, općenito poznat kao „specifikacija dizajna“. U ovoj fazi pripremaju se dizajn mreže, baze podataka, aplikacije, sučelja sustava, korisnička sučelja, dizajn sustava i softvera prema spomenutoj specifikaciji softverskih zahtjeva. Nadalje, tehnički arhitekti i programeri razvijaju logički plan sustava koji zatim pregledavaju svi dionici (www.learntek.org). Ukratko, u ovom trenutku pozivaju se zahtjevi iz SRS dokumenta za stvaranje softverske arhitekture čemu slijedi prevođenje dizajna u izvorni kod.

U fazi razvoja, programeri počinju kodirati prema zahtjevima i razvijenom dizajnu. Ovo je vrijeme kada programeri softvera ulaze i implementiraju kod. Razvoj softvera se često dijeli na manje zadatke koji se raspoređuju između različitih timova s obzirom na njihove vještine (Leau i sur., 2012.). Nakon što je razvojni tim počeo kodirati, puštaju module.

Potom, u fazi testiranja slijedi testiranje spomenutih modula gdje se otkrivaju potencijalni problemi i greške, a programerima se dodjeljuju područja za testiranje. Testeri upućuju na SRS dokument kako bi potvrdili da softver odgovara očekivanjima klijenta. Taj se proces nastavlja sve dok se softver ne usavrši (www.upwork.com). Ova faza se često preklapa sa fazom razvoja kako bi se osiguralo da su svi problemi uočeni i riješeni na vrijeme (Leau i sur., 2012.). Tu se stoga, priprema softver za pokretanje i rad u različitim okruženjima.

Sljedeća faza je faza kontroliranja. Kontroliranje je ustvari osiguravanje da se projekt izvodi u skladu s planom. Tu se dakle kontroliraju odrednice poput kvalitete, opsega, proračuna, rasporeda i rizike. Faza izvršenja i faza kontroliranja ovise jedna o drugoj jer se ustvari odvijaju istovremeno. Dok tim izrađuje softver, voditelj projekta kontrolira njihov rad determinirajući

kvalitetu, troškove i ostale već spomenute čimbenike. Ukoliko postoje neke devijacije od plana, projektni menadžer je obvezan ispraviti iste (Luckey, Phillips 2006.).

Puštanje softvera označava završetak i zatvaranje projekta. Prema Lucky i Phillips (2006.), za projektne menadžere to uključuje provjere financija, finaliziranje dokumentacije te predaju softvera kupcu za posljednje odobrenje kao i upute korisniku za korištenje.

U konačnici, faza održavanja podrazumijeva ažuriranje i podršku softvera čak i nakon isporučenja istog na tržište (ncube.com). Prema Čubranić i sur. (2013.), održavanje softvera predstavlja djelatnost koja podupire operacije sustava u ciljanom okružju osiguravajući potrebna poboljšanja, proširenja, popravke, izmjene i dr.

3.2.2. Metodologije

Razvijene su brojne metodologije koje stavljaju naglasak na različite pristupe prilikom izvođenja projekta za razvoj softvera. Prema Despi (2014.), metodologije razvoja softvera su skup pravila i smjernica koji se koriste u procesu istraživanja, planiranja, projektiranja, razvoju, testiranja, postavljanja i održavanja softverskog proizvoda. Metodologije također uključuju temeljne vrijednosti koje podržavaju projektni tim i alate koji se koristi u planiranju, razvoju i procesu implementacije. Prema Rehmanu i Hussainu (2007.), metodologije pružaju strateški plan razine za upravljanje i kontrolu IT projekta. To je kombinacija međusobno povezanih procesa koji govore što treba učiniti za uspješan projekt. No s druge strane, samo jedna metodologija razvoja softvera ne može se primjenjivati na čitavom opusu raznovrsnih projekata. Sve metodologije i nisu primjenjive u svim projektima. Iz tog razloga, projektni menadžment morao bi prepoznati i utvrditi specifičnu prirodu projekta te prema tome odabrati najprikladniju razvojnu metodologiju. Analogno tome pojavila se potreba za novim, agilnim metodologijama i za procesno-orijentiranim metodologijama razvoja softvera (Čubranić i sur., 2013.).

Metodologije razvoja softvera dijele se u dvije temeljne skupine:

- Heavyweight methodologies tj. teške i poznate kao tradicionalne metodologije razvoja softvera – sadrže mnogo pravila, načina postupanja i dokumentacije, a traže vrijeme i disciplinu za ispravno slijeđenje. Usprkos tome, jednostavne su za razumjeti i provesti. S teškim metodologijama projektni menadžer može lako izvesti praćenje, vrednovanje i izvještavanje. Takve metodologije su pogodne za projekte gdje se zahtjevi rijetko

mijenjaju i gdje složenost projekta dozvoljava detaljno planiranje (Despa, 2014.). Kao najpoznatije teške metodologije ističu se Waterfall (Vodopad) i Spiral Model (Spiralni model) (Khan, Qurashi, Khan, 2011.).

- Lightweight methodologies tj. lagane i poznate kao agilne metodologije razvoja softvera (iako uključuju i ostale metode van agilnog upravljanja projektima) – obično sadrže tek nekolicinu pravila i načina postupanja koji se lagano slijede. Lagane razvojne metodologije prihvaćaju prakse koje programerima omogućuju bržu i učinkovitiju izradu rješenja, uz bolji odaziv na promjene u poslovnim zahtjevima. Svoj fokus uglavnom bazira na razvoju, temeljenom na kratkim životnim ciklusima (isporuka softvera u više uzastopnih iteracija), uključuje kupca, teži jednostavnosti te cijeni ljude (Khan, Qurashi, Khan, 2011.). Nekima od najpoznatijih laganih metodologija smatraju se Scrum, XP, RAD, Prototyping i Lean.

S obzirom na dinamično okruženje u kojem se Hrvatska IT i ICT industrija nalazi i za potrebe ovoga rada, razjasnit će se koja metodologija donosi najveću prednost u kojem kontekstu razvoja softvera. To će se opisati uz pomoć Cynefin matrice. To je okvir koji ocrtava pet situacijskih domena koje su definirane uzročno-posljedičnim odnosima i pomaže voditeljima projekta da shvate kako je svaka situacija drugačija i zahtjeva jedinstven pristup donošenju odluka (www.mindtools.com). Slika 4 prikazuje spomenuti okvir.

Slika 4 Cynefin okvir

Izvor: O'Connor, Lepmets, (2015.)

U jednostavnoj domeni, uzorak i odnosi napora postoje, predvidljivi su, ponovljivi, sami po sebi razumljivi i mogu se odrediti unaprijed. Model odlučivanja temelji se na jedinstvenoj

najboljoj praksi i najprikladniji put akcije je osjetiti – kategorizirati – odgovoriti: komunikacija nije prisutna, samo se informacije emitiraju, a mreža odnosa nije bitna (Pirozzi, 2018.). Stoga, pretpostavka je kako je *Vodopadni model* prikladan izbor, gdje je fiksni proces razvoja softvera popraćen definiranim uzastopnim koracima, bez inkrementalne isporuke, i moguće niske razine iteracije ili zahtjeva za promjenu (O'Connor, Lepmets, 2015.).

Turbulencija i nedostatak bilo kakve poveznice između uzroka i posljedice karakteriziraju domenu kaosa. U nedostatku bilo kakvog pravog odgovora model odlučivanja mora biti da se prvo djeluje, a zatim osjeti i odgovori kao, na primjer, u kriznom upravljanju (Childs, McLeod, 2013.). S obzirom na visoke razine rizika koje su povezane s novim praksama, prikladnim se smatra *Spiralni model* zbog svog pristupa rizičnijim životnim ciklusima (O'Connor, Lepmets 2015.).

Treća domena nalazi se u središtu okvira i zove se domena poremećaja. Ova se kategorija odnosi na situacije u kojima se ne zna koji od ostalih domena primijeniti. Ovo može biti jako teško mjesto jer se ne zna kako „osjetiti" situaciju. U ovoj domeni ljudi djeluju na temelju načina gašenja požara, tj. karakteristično je ponašanje u skladu s osobnim preferencijama i konstantno ispravljanje situacije. Pristup se svodi na razbijanje situacije na manje probleme i zatim primjenjivanje istih u jednu od četiri domene (txm.com).

U kompliciranoj domeni postoji logičan odnos između uzroka i posljedice, ali to nije samo po sebi razumljivo i stoga zahtijeva stručnost. Analitička metoda je potrebna za rješavanje problema ili se za isto poziva stručnjak. Stoga je model odlučivanja osjetiti – analizirati - reagirati. Ovo je domena „dobre prakse" (Puik, Ceglarek, 2015.). Također, smatra se područjem gdje bi ljudi i tim trebali biti u fokus, a ne određeni proces. Stoga se osnaživanje tima može smatrati većom važnosti i prema tome *agilna metoda* s jakim naglaskom na osnaživanju i vlasništvu tima smatra se prikladnom za ovu domenu (O'Connor, Lepmets, 2015.).

Kompleksna domena je područje „nepoznatih znanja", što znači da postoji svjesnost o slabo shvaćenim relevantnim aspektima povezanim s problemom i da je jako malo informacije o samom problemu dostupno (Fierro i sur., 2018.). Ovaj model stoga razvija „sonde" za otkrivanje pojavnih obrazaca (istinski angažman vještog tima omogućava brzo određivanje prioritetnih područja i shodno tome radnji). Također prema French (2013.), uzrok i posljedica u ovoj domeni mogu često su utvrđeni nakon događaja. S obzirom da projekt proizlazi iz interakcije agenata potrebno je procjenom relevantnih informacija „osjetiti" koje su korisne inicijative kako bi se „odgovorilo" pojačavajući ih i osiguravajući ih (Van Beurden i sur.,

2011.). Nadalje, ovu domenu karakterizira sinergija ljudi, iteracije, otvorenost i inovativnost u rješavanju problema i internalizacija cilja u donošenju odluka, koje procesni modeli ne pokrivaju. Stoga primjetno je kako bi *agilne metode poput Scrum okvira* bile prikladan izbor, tj. najbolje rješenje za kompleksne domene (O'Connor, Lepmets, 2015.).

Sve metodologije nastoje svojim metodama pokriti što više faza u razvoju softvera. Neovisno o metodologiji koja se implementira, osnovni motiv svake je razviti kvalitetan softver uz minimalne troškove. Njihov značaj je izuzetan jer uspješno selektirana i upotrjebljena metodologija indicira kvalitetu projekta, i u konačnici samog softvera. S obzirom na pojašnjenje priložene matrice, uočljivo je kako je Scrum okvir za agilno upravljanje projektima pravo rješenje prilikom odabira metodologije za razvoj softvera i to prvenstveno zbog izazovnosti i kompleksnosti okruženja koje ga karakterizira.

3.3. Pregled ključnih izazova upravljanja projektima razvoja softvera

Softverska industrija je vrlo složena i zahtijeva zaposlenike sa specifičnim vještinama, kao i potrebnu stručnost u razvoju softvera. Identificiranje relevantnih ljudi s odgovarajućom razinom znanja za razvojne projekte nije lak zadatak. Industrija softvera također predstavlja jednu od industrija koje se nevjerojatnom brzinom razvijaju, stvarajući okruženje u kojemu poduzeća mogu propasti onoliko brzo koliko su započela, zbog domaće i međunarodne konkurencije. Vlasnici, rukovoditelji, srednji menadžment i svi drugi zaposlenici koji rade u ovom području stalno su pod pritiskom da budu u toku, a stručnjaci za upravljanje projektima pod još su većim pritiskom da osiguraju uspješno izvršavanje projekata. Nadalje, nije dovoljno znati samo o upravljanju projektima. Voditelji projekata također moraju držati korak s ovom industrijom koja se brzo razvija kako bi predvidjeli mogući rizik, kvalitetu, integraciju, financijske i druge čimbenike koji mogu ometati šanse za uspješan projekt. Ti se čimbenici mogu primijeniti na mnoge industrije, ali zbog brzine kojom se tehnologija mijenja, a konkurencija povećava pritisak da se projekti isporuče na vrijeme, u skladu s proračunom, te s očekivanim standardima kvalitete, upravljanje projektima u softverskoj industriji čini izrazito teškim zadatkom. Slijede ključni izazovi upravljanja projektima razvoja softvera biti će raspoređeni u tri skupine radi veće preglednosti.

3.3.1. Izazovi vezani uz korisnika

Uključenost korisnika u proces razvoja softvera je vrlo važan aspekt za uspjeh projekta. Brojne metodologije navode da bi kupac trebao biti dio razvojnog procesa od analize i dizajna pa sve do implementacije i održavanja. Međutim, prema Cho (2010.), otkriveno je kako programeri imaju poteškoće u radu s klijentima na softverskim projektima. Voditelj projekta istaknuo je nedovoljnu uključenost korisnika tj. njihovu ne prisutnost sve do okončanja projekta. Osim toga primijećeno je kako su korisnici često zauzeti i okupirani drugim poslovima i nemaju vremena razgovarati s programerima cijeli dan. To u konačnici smanjuje komunikaciju s korisnikom od jednom tjedno do samo dva puta po Sprintu. Kao sljedeći veći izazov izdvojeno je nedodjeljivanje programerima specifikacijske dokumentacije, to znači produžiti vrijeme projekta dodatno za izradu istih i prenesena odgovornost o specifikacijama na programere. Nadalje, uočeno je kako, većinu vremena, kupci ne znaju što zaista žele u svom budućem sustavu i to postaje prepreka za uključivanje korisnika u proces razvoja projekta. Prema Cho (2010.), pokazalo se kako kupci misle da imaju jasnu ideju o softveru ali često se ispostavi da nemaju, da daju manjak informacija što se u konačnici može svesti na nejasne zahtjeve korisnika. To stavlja izazov na programere da shvate što to točno korisnici traže jer inače nemaju dovoljno informacija za daljnji rad. Osim toga, Han i Huang (2007.) prepoznali su još nekolicinu izazova vezanih za korisnike prilikom upravljanja projektima za razvoj softvera. To su sljedeći – korisnici često znaju biti otporni na promjene, pojavljivanje sukoba među korisnicima, pojedini korisnici imaju negativne stavove prema projektu i manjak predanosti projektu.

3.3.2. Izazovi proizašli iz složenosti projekta

Softverski projekti jedni su od najsloženijih pothvata današnjice. Povećana složenost dovela je do velikog broja neuspjeha softverskih projekata u smislu vremena, kvalitete, troškova itd. Glavni razlog povećane složenosti je to što sve dok razvoj softverskog projekta ne završi, softver ostaje nevidljiv, a sve što je nevidljivo, time je teško upravljati i kontrolirati. Izuzev toga, složenost projekta determinira više čimbenika, a svi oni u konačnici predstavljaju izazov za upravljanje projektima za razvoj softvera. Prva determinanta očituje se u složenosti upravljanja vremenom. Dakako to uključuje procjenu trajanja projekta kao i izradu rasporeda za različite aktivnosti i pravovremeni završetak projekta. Ukoliko je projekt od velike složenosti, često će doći do zaostajanja, što mijenja raspored projekta i podrazumijeva kasnu isporuku softvera (www.geeksforgeeks.org). Nadalje, u velikim i složenim projektima često

dolazi i do izazova vezanih uz troškove projekta. Procjena ukupnih troškova projekta izrazito je zahtjevan zadatak, a kod takvih projekata nije malo vjerojatno da će se dogoditi neočekivano proširenje opsega što će voditi ka prekoračenju budžeta i u većini slučajeva neuspjehom cijelog projekta (www.geeksforgeeks.org). Kao posljednji čimbenik koji čini softverske projekte teškima za upravljanje vezan je uz tehnologiju. U softverskim projektima, tehnologija je ključni atribut, no u situacijama kada su projekti nepredvidivi i složeni često se zna dogoditi nešto od sljedećeg – projekt uključuje novu tehnologiju, postoji visoka razina kompleksnosti određene tehnološke opreme i ne može bilo tko sa njome rukovoditi, tehnologija je zastarjela ili „nezrela“ te korištenje tehnologije koja nije bila korištena u dosadašnjim projektima (Han, Huang, 2007.).

3.3.3. Izazovi vezani uz tim

Kako u klasičnim, tako i u projektima za razvoj softvera neizbježan je pojam međuljudskog sukoba. Nekoliko čimbenika doprinosi takvim sukobima poput rada pod visokim pritiskom performansi, različitog razmišljanja, razlika u poslovima te različite kulture zaposlenika. Ako se njima ne kontrolira i učinkovito upravlja, ti bi sukobi mogli negativno utjecati na produktivnost i na taj način ometati cjelokupni projekt. Stoga, nedostatak osoblja ističe se kao prvi izazov iz ove domene koja može negativno utjecati na upravljanje projektom. Potreban je odgovarajući broj ljudi s valjanim iskustvom i znanjem u adekvatnim područjima (Boehm, 2006.). To upućuje na sljedeći problem, a to je neadekvatno osposobljen tim za razvoj softvera. Takvo nešto može upućivati ili na neiskustvo pojedinih članova ili na nedostatak pružene specijalizacije za usvajanje potrebnih vještina koje zahtijevaju ovakvi projekti (Han, Huang, 2007.). Pored navedenog, poznato je da je neučinkovita komunikacija korijen većine grešaka u softverskim projektima. Nastavno na to, programeri softvera često se bore s pronalaskom smislenih izraza, jezika, alata i artefakata kako bi uspjeli u komunikaciji s ostalim članovima tima (Dey i sur., 2016.). Takav problem, tj. manjak komunikacije između timova, mogao bi uzrokovati probleme poput dupliciranog posla ili neprimitka informacija krucijalnih za razvoj projekta (Cho, 2007.). Također, prevelika samouvjerenost ili manjak znanja kod projektnih menadžera u timu jedan je od značajnih izazova u ovakvim projektima. Drugim riječima, mnogi softverski projekti propadaju jer projektni menadžeri uglavnom planiraju na temelju svog iskustva gdje nedostaju znanstvene metode koje bi ih podržale. (Huynh, Nguyen, 2020.). Osim toga, u današnjici česta je pojava rada u virtualnim timovima koji također rezultiraju posebnim setovima problema za softverske projekte. Novi set izazova s kojima se tim susreće u takvim okolnostima najčešće su manjak spoznaje o tome što druge kolege rade i pad koncentracije

prilikom „brainstorminga“. Uočilo se kako su virtualni timovi na taj način rada smanjili produktivnost zaposlenika pretežito zbog promjena u pogledu manjka društveni aktivnosti, nedostatka razgovora ne vezanih uz posao, prije i nakon posla, drugim riječima zbog manjka van poslovnog povezivanja tima (Miller i sur., 2021.).

Pored ovih glavnih izazova koji utječu na upravljanje projektima razvoja softvera važni spomena su također neki izazovi vezani uz organizacijsku okolinu poput promjene menadžmenta za vrijeme izvođenja projekta, restrukturiranje organizacije, nestabilna organizacijska okolina ili pak negativni utjecaj korporativnih politika na projekt (Han, Huang, 2007.). Naravno neizostavno je za spomenuti potencijalne izazove koji se mogu pojaviti direktno iz procesa kreiranja softvera kao što su neispravna integracija modula, greške sučelja, loša izrada prototipa softvera, pogrešni scenariji i manjkavosti koda (Kilibarda i sur., 2016.). Također, pokazalo se da izazovi prilikom testiranja softvera često dovode do loše kvalitete specifikacija testnih slučajeva, a time i negativno utječu na vrijeme, troškove i vjerojatnost otkrivanja nedostataka (Juhnke i sur., 2020.). Primjetni su i izazovi trenutne pandemije zbog koje nastaju direktne implikacije na dizajn, razvoj i samu upotrebu tehnologija za razvoj softvera (He, Zhang, Li, 2020.). Svi ovi izazovi zahtijevaju adekvatni nadzor i pravovremeno rješavanje kako bi upravljanje projektima razvoja softvera moglo proći bez većih poteškoća.

4. EMPIRIJSKO ISTRAŽIVANJE IZAZOVA UPRAVLJANJA PROJEKTIMA RAZVOJA SOFTVERA PRIMJENOM SCRUM OKVIRA

Kako bi se opisani teorijski okvir ovoga rada povezao s poslovnom praksom i kako bi se ostvarili postavljeni ciljevi, provedeno je empirijsko istraživanje putem ankete, istražujući izazove s kojima se susreću poduzeća koja se bave softverskim projektima prilikom korištenja Scrum okvira. Više informacija o anketi i njezinoj analizi izneseno je u naslovu Rezultati istraživanja.

4.1. Metoda istraživanja

Za istraživanje se koristila Grounded theory approach tj. utemeljena teorija koja se bavi generiranjem teorije i bazirana je na podacima koji su sustavno prikupljeni i analizirani (kvalitativno istraživanje). Drugim riječima, ovaj rad uključuje izgradnju hipoteza i teorija kroz prikupljanje i analizu podataka. Stoga, kako bi se istražili izazovi, inicijalno je bilo potrebno ispitati primjenjivost Scrum okvira u IT i ICT poduzećima u Republici Hrvatskoj. Nakon telefonskih poziva i mailova upućenih u 120 poduzeća, uzorak poduzeća koji koriste Scrum okvir suzio se na 28 poduzeća. Nadalje, temeljem vrlo intenzivnog sekundarnog istraživanja sastavljena je anketa. Ista je napravljena u Google obrascu i proslijeđena elektronskim putem u spomenutih 28 poduzeća. Većina poduzeća anketu je ispunila jedanput, dok je nekolicina poduzeća ispunila više puta (osobe drugačijih uloga su ispunjavale anketu – Scrum Master, tester, developer, Vlasnik proizvoda i sl.). Zbog takvog pristupa iz ankete je dobiveno 40 odgovora.

U prvom dijelu ankete, cilj je bio sakupiti opće informacije o poduzećima kako bi se dobio bolji uvid o tome tko, tj. kakve organizacije sačinjavaju bazen ispitanika. To je uključivalo od ispitanika da odaberu opciju koja opisuje veličinu njihovog poduzeća, definiraju koliko dugo u svome poslovanju primjenjuju Scrum okvir prilikom upravljanja projektima i primjenjuju li ga u potpunosti ili samo dijelove te da definiraju ili odaberu opciju koja opisuje njihovu ulogu u timu.

Drugi dio ankete bio je usmjeren na potencijalne izazove sa kojima su se poduzeća susretala prilikom korištenja Scrum okvira za agilno upravljanje projektima. Za početak se od

ispitanika zahtijevalo da od 39 ponuđenih izazova označe sve s kojima su se do sada susretali. Nadalje, od ispitanika se zatražilo da opisnim odgovorom pobliže objasne izazove sa kojima su se susretali i da razlože zbog čega do njih dolazi. Kao posljednje pitanje, ispitanicima se ponudilo da navedu dodatne izazove koji su možda bili previđeni od strane autorice i da ih ukratko objasne ukoliko isti postoje.

4.2. Rezultati istraživanja

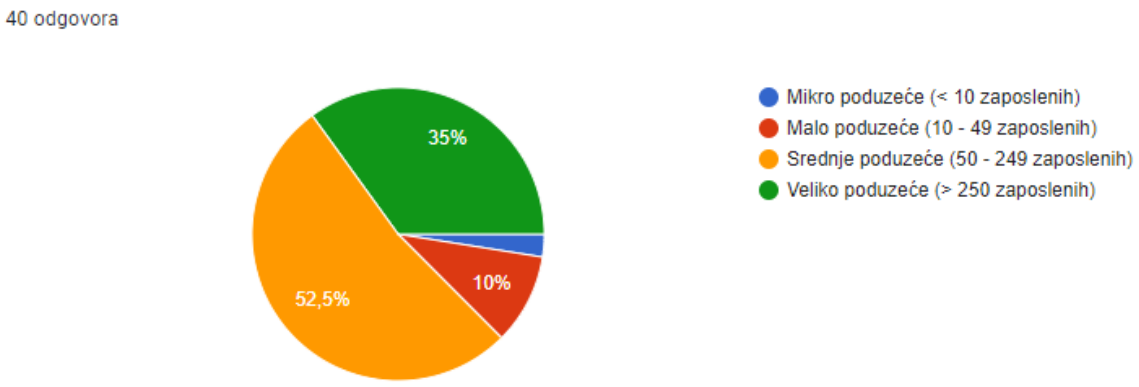
U ovom dijelu poglavlja slijedi analiza dobivenih rezultata istraživanja. Analiza odgovora će se obrađivati kroz objašnjavanje svakog pitanja pojedinačno i potom njegovih odgovora.

Za početak, od ispitanika se zatražilo da odaberu opciju koja opisuje veličinu njihovog poduzeća. Opcije koje su se nudile bile su:

- Mikropoduzeće (<10 zaposlenih)
- Malo poduzeće (10 – 49 zaposlenih)
- Srednje poduzeće (50 – 249 zaposlenih)
- Veliko poduzeće (>250 zaposlenih)

Podaci su pokazali kako je na anketi sudjelovalo samo jedno mikro poduzeće (2.5 %), dok malo poduzeće čini 10 %, srednje 52.5 % i u konačnici veliko 35 %. Veličina poduzeća relevantan je čimbenik jer je zanimljivo promatrati s koliko istih izazova se ustvari susreću poduzeća usprkos različitoj veličini. Spomenuto je vidljivo na slici 5.

Slika 5 Rezultati ankete – grafikon prikaza veličine poduzeća



Izvor: djelo autorice rada prema rezultatima primarnog istraživanja

Nadalje, ispitivanjem koliko se dugo primjenjuje Scrum unutar odabranih poduzeća dobiveni su rezultati prikazani tablicom 1. Tablica prikazuje kako je prosječan broj godina primjenjivosti Scrum okvira unutar različitih poduzeća četiri godine, dok ga najkraće koristi jedna organizacija u dužini od tri mjeseca, a najveći period je 12 godina. Istraživanjem se pokazalo kako se pojedini isti izazovi pojavljuju u organizacijama koje su nove u primjenjivanju Scrum okvira ali i kod onih kojima je Scrum već dugo godina u praksi korištenja. Točnije, to su izazovi poput ograničenog znanja o domeni od strane Scrum tima, nejasne obveze te nedostatak Scrum obuke.

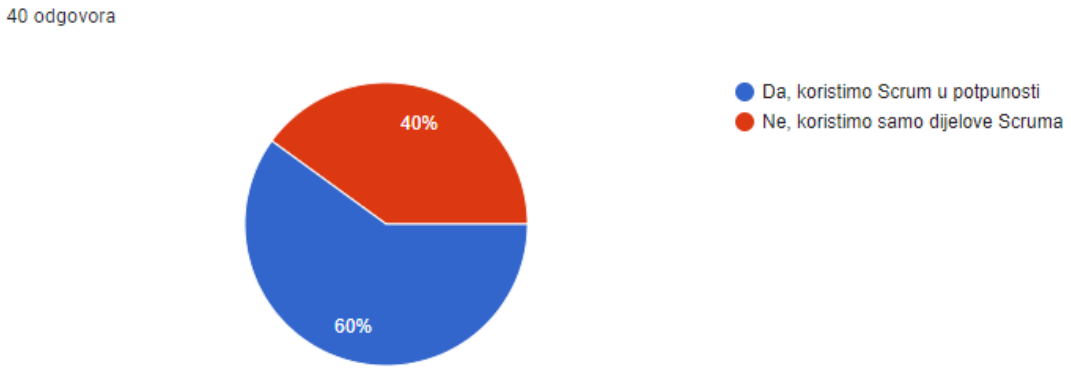
Tablica 1 Rezultati ankete – prikaz dužine korištenja Scrum okvira

Dužina korištenja	Broj odgovora
3 mjeseca	1
4 mjeseca	1
9 mjeseci	1
10 mjeseci	1
1 godina	1
1 godina i 1 mjesec	1
1.5 godina	2
2 godine	6
3 godine	4
4 godine	6
4 godine i 3 mjeseca	1
5 godina	6
6 godina	3
6 godina i 9 mjeseci	1
7 godina	1
8 godina	1
9 godina	1
10 godina	1
12 godina	1

Izvor: djelo autorice rada prema rezultatima primarnog istraživanja

Nadalje od ispitanika se zatražilo da odaberu i pojasne koriste li Scrum u potpunosti ili samo pojedine dijelove. Pokazalo se kako čak 40 % ispitanih poduzeća ne koristi u potpunosti Scrum okvir. Ostalih 60 % je reklo kako koristi Scrum u potpunosti prilikom upravljanja softverskim projektima unutar svoga poduzeća. Ti podaci vidljivi su na slici 6. Nadalje, slijede odgovori ispitanika o tome koje dijelove koriste i isto tako i obrazloženja zašto koriste samo pojedine dijelove Scrum okvira.

Slika 6 Rezultati ankete – grafikon prikaza korištenja Scrum okvira



Izvor: djelo autorice rada prema rezultatima primarnog istraživanja

Istraživanjem je otkriveno da neka poduzeća koriste samo Daily Scrum, planiranje sprinta, user Story, backlogs dok drugi pak samo planiranje, Daily Scrum, retrospektive te Backlog kao komunikaciju – razvoj i određivanje prioriteta. Klasično granuliranje i sprintove samo za velike projekte. Također, značajan dio odgovora bilježi da poduzeća često prilagađaju Scrum okvir prilikom upravljanja projektima prema svojim ili korisnikovim potrebama. Neki od objašnjenja su sljedeći:

„Prilagodili smo svojim okvirima i potrebama. Retrospektiva i Sprint review nisu obavezni nakon svakog sprinta, te duljina sprinta nije uvijek identična.“

„Koristimo uloge, artefakte, skoro sve ceremonije. Nešto smo prilagodili sebi (npr. ponekad product increment ne mora biti gotov unutar jednog sprinta ako tako na planiranju dogovorimo i sl.)“

„Koristiti Scrum u potpunosti uzima više vremena za administraciju za koji jednostavno nemamo ljudi da se bave s tim.“

„Koristimo Scrum ceremonije (daily standup, Sprint planing, Sprint review). Nije uvijek primjenjivo u praksi koristiti Scrum u potpunosti, pogotovo kada je scope i timeline jasno definiran.“

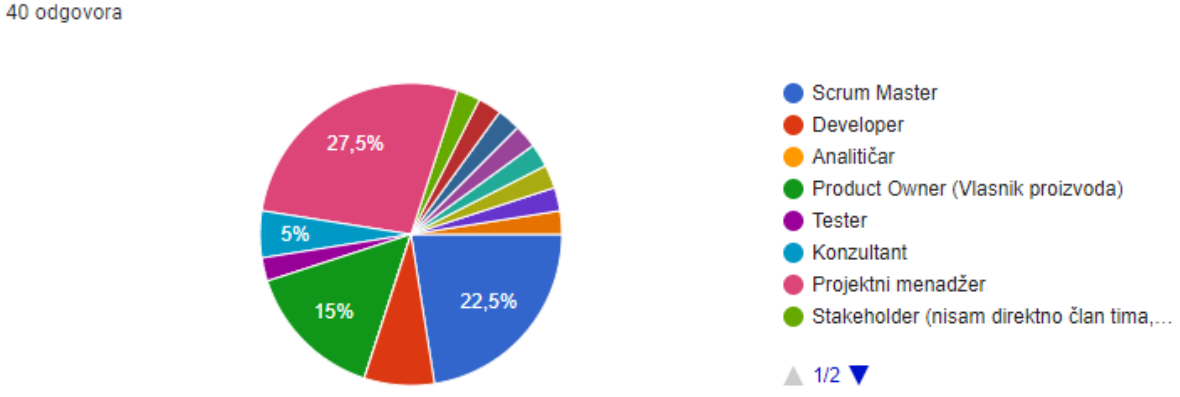
„Scrum u potpunosti može biti uveden samo ako je cijela organizacija tako postavljena interno, ali i ako su klijenti spremni na takav rad. Još uvijek, u najvećem postotku klijenti očekuju waterfall isporuke i planiranje. U pravilu se koristi neki hibridni Scrum u kojem se uzima maksimalan broj elemenata: uobičajeno je to daily standup (to imaju i oni timovi koji ništa

drugo ne rade po Scrumu), ostale ceremonije (grooming, planing..). Ovdje pokušavamo i User storye pisati po pravilima (As a user... I want..so that..). Ono gdje je Scrum najteže uvesti su iterativne isporuke i tu se i dalje uglavnom koristi fiksni opseg koji se zna n mjeseci unaprijed i samo se po fazama slaže.“

Prema priloženim objašnjenjima vidljivo je kako brojna poduzeća uspijevaju isporučiti dobar softver ne koristeći Scrum okvir u potpunosti. Primjetno je kako mnogi smatraju, ne samo da nisu ni potrebni svi dijelovi za uspješan projekt, nego da negdje jednostavno nisu ni mogući. Prilagođavanje okvira prema svojim potrebama pojedinim organizacijama ubrzava proces, uklanjajući određenu administraciju kao i sastanke nakon sprinteva kako bi dobili na vremenu. Također, fiksni opseg i fiksni vremenski okvir se teško uvodi i prati te stvara ponajveće izazove u primjeni Scrum okvira pa ga pojedina poduzeća izmjene prema potrebi ili jednostavno ne implementiraju kako bi lakše provodili projekt. Osim toga, često želje, potrebe i neznanje klijenata tjeraju poduzeća da mijenjaju rad u Scrum-u, koristeći kombinirane pristupe koji su njima više poznati unatoč dobroj funkcionalnosti potpunog Scrum okvira za agilno upravljanje projektima.

Potom, ispitanici su bili upitani da od ponuđenih odgovora odaberu koji opisuje njihovu ulogu u projektnom timu. Također, nudila im se opcija da dopišu svoju ulogu ukoliko ista nije navedena u popisu. Uloge koje su se nudile su Scrum Master, developer, analitičar, Product Owner (Vlasnik proizvoda), tester, konzultant i projektni menadžer. Kao nadopisane uloge pojavile su se Stakeholder (uz navedeno da ispitanik nije direktno član projektnog tima, ali je dio menadžmenta koji je odgovoran za metodu po kojoj rade i isporučuju vrijednost klijentima), voditelj razvoja, voditelj inženjeringa te kombinirane uloge već navedenih pozicija. To je u postotcima prikazano na slici 7.

Slika 7 Rezultati ankete – grafikon prikaza uloga u projektnom timu



Izvor: djelo autorice rada prema rezultatima primarnog istraživanja

Vidljivo je kako je najveći postotak odgovora došao od projektnih menadžera koji čine čak 27.5 %. Sljedeći po redu je Scrum Master s 22.5 %, zatim Product Owner s 15 %, pa developer sa 7.5 %. Ulogu konzultanta odabrale su dvije osobe, dok su ostale uloge odabrane po jedanput. Uloge ispitanika relevantne su u ovome istraživanju jer će se pojedini izazovi iz ankete promatrati kroz njihove odgovore.

Sljedeće, u anketi je bilo navedeno 39 izazova sa kojima su se poduzeća mogla susretati prilikom korištenja Scrum okvira za agilno upravljanje softverskim projektima. Izazovi koji su bili ponuđeni idu redom ovako: Zahtjevi klijenta se prečesto mijenjaju; Product Owner (Vlasnik proizvoda) nije dovoljno prisutan / dostupan; Manjak komunikacije; Ljudi koji pišu kod ne surađuju dovoljno s ljudima koji ga testiraju; Nedostatak Scrum obuke; Ograničeno znanje o domeni od strane Scrum tima; Loše planiranje projekta; Manjak vremena ili neodgovarajuća raspodjela vremena; Nejasne obveze; Neadekvatna radna okolina (problem privatnih ureda ili otvorenog prostora); Nedostatak tj. neodržavanje retrospektivnih sastanaka nakon svakog sprinta; Prekoračenje budžeta; Nedostatak resursa; Nedostatak detaljne dokumentacije; Kompleksnost projekta; U Scrum-u ne postoji plan ili strategija za rješavanje rizika; Zastarjela tehnologija; Visoka razina kompleksnosti određene tehnološke opreme sa kojom ne može svatko rukovoditi; Nedostatak jasnoće u Daily Scrum-u; Daily Scrum sastanci oduzimaju previše vremena; Nedovoljno dobra priprema Product Backloga – „nespreman“ Product Backlog; Izazovi oko dodjeljivanja prioriteta PBI-evima (Product Backlog Items); Testiranje nije dovršeno do kraja sprinta; Provođenje sprintova kao „mini-waterfalls“; Neki Scrum timovi rade na sprintu različite duljine od drugih; Sprintovi su prekratki da bi pružili smislene značajke; Obično ima nekoliko nepotpunih „storya“ na kraju sprinta; Previše „story-a“ počinje prerano; Nemogućnost pisanja malih „storya“; Kriteriji prihvaćanja nisu dobro poznati za mnoge „storye“; Korištenje „storya“ prekida kontinuitet; Neispravna integracija modula; Greške sučelja, Pogrešni scenariji, Manjkavosti koda, Pojavljivanje mnogo „bugova“, Izazovi kod upravljanja opsegom; predugački razvojni ciklusi te u konačnici Prototip se ne može isporučiti. Rezultati su prikazani u tablici 2.

Tablica 2 Rezultati ankete – tablica prikaza izazova

IZAZOV	POSTOTAK ODGOVORA
Zahtjevi klijenta se prečesto mijenjaju	45%
Product Owner (Vlasnik proizvoda) nije dovoljno prisutan / dostupan	47,50%
Manjak komunikacije	40%
Ljudi koji pišu kod ne surađuju dovoljno s ljudima koji ga testiraju	27,50%
Nedostatak Scrum obuke	45,00%
Ograničeno znanje o domeni od strane Scrum tima	27,50%
Loše planiranje projekta	30,00%
Manjak vremena ili neodgovarajuća raspodjela vremena	55,00%
Nejasne obveze	30,00%
Neadekvatna radna okolina (problem privatnih ureda ili otvorenog prostora)	5,00%
Nedostatak tj. neodržavanje retrospektivnih sastanaka nakon svakog sprinta	15,00%
Prekoračenje budžeta	17,50%
Nedostatak resursa	47,50%
Nedostatak detaljne dokumentacije	27,50%
Kompleksnost projekta	37,50%
U Scrum-u ne postoji plan ili strategija za rješavanje rizika	5,00%
Zastarjela tehnologija	2,50%
Visoka razina kompleksnosti određene tehnološke opreme sa kojom ne može svatko rukovoditi	2,50%
Nedostatak jasnoće u Daily Scrum-u	5,00%
Daily Scrum sastanci oduzimaju previše vremena	12,50%
Nedovoljno dobra priprema Product Backloga - "nespreman" Product Backlog	50,00%
Izazovi oko dodjeljivanja prioriteta PBI-evima (Product Backlog Items)	27,50%
Testiranje nije dovršeno do kraja sprinta	32,50%
Provođenje sprintova kao "mini-waterfalls	30,00%
Neki Scrum timovi rade na sprintu različite duljine od drugih	2,50%
Sprintovi su prekratki da bi pružili smislene značajke	12,50%
Obično ima nekoliko nepotpunih "storya" na kraju sprinta	40,00%
Previše "story-a" počinje prerano	5,00%
Nemogućnost pisanja malih storya	17,50%
Kriteriji prihvaćanja nisu dobro poznati za mnoge "storye"	12,50%
Korištenje "storya" prekida kontinuitet	0,00%
Neispravna integracija modula	5,00%
Greške sučelja	7,50%
Pogrešni scenariji	7,50%
Manjkavosti koda	17,50%
Pojavljivanje mnogo „bugova“	35,00%
Izazovi kod upravljanja opsegom	27,50%
Predugački razvojni ciklusi te u konačnici	10,00%
Prototip se ne može isporučiti	25,00%

Izvor: djelo autorice rada prema rezultatima primarnog istraživanja

Primjetno je kako je izazov s najvećim brojem glasova manjak vremena ili neodgovarajuća raspodjela vremena i on čini čak 55 % ispitanih. Prema dobivenim odgovorima iznenađujuće je kako je taj izazov čest pokretač brojnih drugih problema prilikom razvoja softvera, ali ujedno i ono očekivano, da je ustvari rezultat drugih izazova:

„U trenutnom timu glavni izazovi nastaju radi manjka vremena u planiranju i organizaciji razvoja, stihijski se pokrenuo razvoj proizvoda i sad cijeli tim (uključivši mene kao PO) jurimo da nešto stignemo. Osim toga, tim je prevelik (15ljudi) i nisu upoznati sa SCRUM-om (znaju neke osnove, tipa kako se story piše - format) zbog čega također gubimo na vremenu.“

„Glavni problem je nedostatak vremena, odnosno raspoloživost ljudi da rade na projektu. Kad se to spoji s kompleksnošću projekta imamo problem koji pokušavamo riješiti forsiranjem. Tad se javlja nespreman backlog, nedovoljna komunikacija, tehnički dugovi itd.“

„Ljudi dolaze ne pripremljeni na sastanke, pa trošimo previše vremena na objašnjavanje neke teme o kojoj su mogli pročitati iz dokumentacije.“

Iz priloženih komentara jasno je vidljivo kako se različita poduzeća susreću sa istim problemom poradi drugačijih razloga. Ishod neadekvatnog upravljanja vremenom prilikom upravljanja softverskim projektima donosi kobne rezultate, a to je u konačnici činjenica da se prototip ne može isporučiti. Zanimljiv je podatak iz ankete kako se od 22 ispitanika koji su označili manjak vremena ili neodgovarajuća raspodjela vremena, njih samo 10 izjasnilo kako se susreću i sa izazovom da ne mogu isporučiti prototip na vrijeme. Može se pretpostaviti dakle, kako poduzeća usprkos manjku vremenu uspijevaju završiti na vrijeme, forsirajući rad i radeći ubrzano što na kraju vrlo vjerojatno dovodi do isporučivanja proizvoda značajno slabije kvalitete.

Kako je već viđeno i u prethodnim objašnjenjima, jedan od čestih skupina izazova sa kojima se organizacije susreću su oni vezani direktno uz područje Scruma, drugim riječima, ograničeno znanje o domeni od strane Scrum tima (27.5 %) i nedostatak Scrum obuke (45 %). Kako bi Scrum okvir uopće mogao funkcionirati primijenjen na neki projekt, važno je da ga osobe u timu znaju implementirati i onda njime upravljati. Često zbog baš tih izazova se poduzeća susreću sa raznim drugim izazovima i problemima u softveru jer rade greške koje bi izbjegli da su pravilno obučeni o Scrum okviru za upravljanje projektima razvoja softvera:

„Po mom mišljenju, 99 % problema (izazova) proizlazi iz nepoznavanja Scruma i njegovim krivim korištenjem. Scrum nisu samo ceremonije i role, nego “mindset” „

„Osnovni problem iz kojeg proizlaze svi ostali izazovi je - nepoznavanje osnova Scrum-a. Svi ga prilagođavaju sebi kako im paše i onda to više nije Scrum nego hibrid i onda dolazi do problema.“

„Product Owneri nisu dovoljno educirani za svoju ulogu i nemaju potrebnu podršku od strane kompanije.“

„Premalo Scrum obuke tj. certificiranih Scrum Mastera zbog nedostatka vremena za pripremu i polaganje certifikata“

„Često se na projekt stavljaju ljudi koji nemaju odgovarajuće znanje te je potrebno mentorstvo koje oduzima vrijeme te je potrebno više vremena da se odrade dnevni zadaci“

„U početku nam je glavni izazov bio nerazumijevanje uloga, ceremonija i samog okvira.“

„Najveći problem je manjak znanja i razumijevanja prema tome što Scrum okvir znači i koja je svrha. Netko je pročitao članak i uveo standardni Scrum okvir i forsira se na svakom projektu.“

„Mlađi članovi tima trebaju dodatne edukacije po pitanju razumijevanja Scruma, story pointova i sl. Uglavnom lako rješivo ali nešto što se često ponavlja.“

Prema dobivenim podacima iz poduzeća o spomenutoj problematici vidljivo je kako neadekvatan i disfunkcionalan trening u području Scrum okvira dovodi do negativnih ishoda i posljedica koje uključuju nerealna očekivanja, te nespremnost i poteškoće pri promjenama. Izostanak dubokog razumijevanja Scrum vrijednosti posljedično vodi i do nedostatka učinkovite suradnje te niskog samopouzdanja članova tima. Nedostatak obuke često ukazuje na opći nedostatak organizacijske posvećenosti procesu. Trening može omogućiti timovima da izbjegnu mnoge pogreške prilikom upravljanja projektima. Pružanje timovima pravi temelj vještina presudno je za uspjeh softverskih projekta.

Nadalje, dobro je poznato kako je neadekvatna komunikacija često razlog propadanja softverskih projekta. U ovoj anketi izazov „manjak komunikacije“ također se pokazao kao jedan od vodećih problema i čini 40 % odgovora.

Sličnosti i razlike sa kojim se susreću poduzeća kod problema komunikacije temeljeni su pretežito na odgovorima dobivenim od strane Scrum Mastera, Developera i Projektnih Menadžera.

„Premalo korištenje asinkrone komunikacije u firmi, gdje se za sve sazivaju sastanci bez ikakvog centralnog mjesta za komunikaciju u kojoj bi se neke stvari mogle asinkrono dogovoriti.“

„Većina izazova proizlazi iz loše komunikacije "product owner-a" prema članovima tima, te činjenice da developeri sve taskove pišu sami.“

„Često nastaju problemi kada developeri međusobno ne komuniciraju dovoljno. Npr. frontend i backend ne rješavaju problematiku zajedničkog zadatka kada rade na tom zadatku, tj. nisu dovoljno temeljiti da provjere je li i druga strana zadatka odrađena. To rezultira time da na daily Scrumu ustanovimo da jedan od ta dva dijela zadatka ne štima, a to su oni već trebali riješiti međusobno. Nadalje, ostali problemi uglavnom nastaju zbog nedovoljne ili nejasne komunikacije. Primjerice ako nešto što je BITNO za softver odjednom ne funkcionira, to se mora na vrijeme i s visokim prioritetom naglasiti / dojaviti timu. A poruka se često izgubi među ostalim dnevnim razgovorima. Član tima je to možda i dojavio pa ne osjeća dodatnu odgovornost, ali ako je svjestan da tim nije dodijelio dovoljnu pažnju tom problemu, onda njegova dužnost nije gotova, on to mora urgirati i nametnuti kao #1 prioritet ili se zasebno javiti PM-u.“

Zanimljivo je primijetiti kako sve tri navedene uloge definiraju izazov komunikacije sa skroz drugačijih perspektiva. Istraživanjem je primijećeno kako loša komunikacija u timu varira od vlasnika proizvoda pa ustvari do svakog člana u timu što dovodi do zaključka kako svaka strana u timu mora znati kako i na vrijeme iskomunicirati problem ili potrebu. Loše prenošenje ključnih poruka čiji je uzrok loša komunikacija posljedično rezultira nizom drugih problema. Također, prilikom analize odgovora uočeno je i osvrtanje na današnje okolnosti sa pandemijom Covid-19 i njegovim negativnim utjecajem na komunikaciju u timu: *„…rad od kuće otežava komunikaciju usprkos Teams/Zoom-u.“*. Potreba za centralnim mjestom za komunikaciju, svakako je rješenje koje bi u konačnici odgovaralo svim stranama.

No, dobra komunikacija unutar tima i dalje se može postići pri korištenju Scrum okvira, ali komunikacija s kupcem može biti problematična. Tako je ustvari, temeljem podataka dobivenih iz poduzeća uočeno kako klijenti sa sobom donose i novi set izazova:

„Događa se da između dvije strane (klijent − razvojni tim) nije jasno objašnjeno što se točno želi dobiti. To rezultira promjenama zahtjeva usred sprintova i nepotrebnim traćenjem vremena. Klijent ponekad ima nerazumne zahtjeve za male vremenske rokove, ne shvaćajući kako je to što je on zatražio zapravo jako kompleksno i iziskuje više vremena za development

nego što bi on htio. Tu često nastaju pritisci na projektnog menadžera i tim, a PM mora mitigirati te pritiske i ublažiti nasrtaj na tim te obrazloženo prodiskutirati koje je najbolje izvedivo i logično rješenje.“

Iako je prethodni odgovor doista dubinski objasnio problematiku koja dolazi sa klijentima, također potrebno je spomenuti i kako se iz detaljnije analize ostalih odgoovra primjećuje isti izazov: *„Zahtjevi klijenta se često mijenjaju. Klijenti ne iznesu svoje mišljenje u potpunosti te onda naknadno pokušavaju promijeniti dogovoreno.“*. Nadalje, ovaj izazov promatran je i iz perspektive kulture organizacije i iznosi sljedeće: *„Izazov je prenijeti kulturu i način rada klijentu, dakle bez njegovog razumijevanja, klijent vas neće moći pratiti, te će samo rezultirati u njegovoj nesigurnosti, čestim izmjenama, te lošim odlukama.“*

Analizirajući odgovore ispitanika primijećeno je kako se organizacije često susreću sa problemima oko klijenta, bilo to prečesto mijenjanje zahtjeva (prema podacima iz ankete čak 46.2 %), nerazumijevanje Scrum procesa ili pak ne spremnost na korištenje Scrum okvira. Također, neaktivno uključivanje klijenta u proces razvoja softvera utječe na promjene plana tijekom sprintova, nezadovoljstvo obiju strana kao i u konačnici potencijalno propadanje projekta. Ovakve probleme ne smije se ignorirati, već njima upravljati te adekvatno educirati klijente o njihovim obvezama i mogućnostima prije početka samoga projekta.

Kao što je definirano u teorijskom dijelu rada, Product Backlog predstavlja Popis stavki koje proizvod treba imati. Nedovoljno dobra priprema Product Backloga , tj. "nespreman" Product Backlog izazov je sa kojim se poistovjetilo čak 50 % ispitanih. Tako je temeljem podataka dobivenih iz poduzeća, ovaj izazov opisan sljedećim izjavama:

„Backlog je vrlo često previše općenit i nedovoljno dobro strukturiran i prioritiziran.“

„Loše pripremljen backlog pa se idući sprint ne može niti isplanirati, niti započeti. Dolazi do toga jer product ljudi budu prezatrpani s poslom u drugim timovima ili ljudskom greškom ili nemarom.“

„Dolazi do toga zato što se Backlog ne puni na vrijeme. Kad se god pojavi zadatak koji treba ići u Backlog, najbolje ga je odmah upisati. Ako se to ostavi za kasnije, najčešće se i ne upiše pa kasnije imamo nepotpun Backlog. Svi znamo da neki zadaci nedostaju, ali se teško prisjetiti koji točno i koliko ih je.“

Iz priloženog je vidljivo na koji način dolazi do nespremnog Popisa stavki koje proizvod treba imate, te što to posljedično uzrokuje. Stoga, potrebo je ovaj izazov minimizirati redovitim

podsjećivanjem tima da zadatke ubacuju u Popis. Također, prije planiranja sprinta voditelj projekta s par ključnih developera trebao bi proći kroz Popis i nadopuniti ga sa detaljima o zadacima ukoliko je potrebno. Štoviše i izazovi oko dodjeljivanja prioriteta stvarima na Popisu stavki za proizvod velik su problem ispitanicima i na anketi čine čak 27.5 %. Osim izazova vezanih uz Popis, rezultati ankete su pokazali kako se izazovi sa kojima su se organizacije često susretale odnose na nejasne obveze (30 %), manjak resursa (47.5 %), kompleksnost projekta (37.5 %) te neprisutnost Vlasnika proizvoda (47.5 %). Iziskuje dosta truda na samom početku projekta kako bi se projekt kvalitetno isplanirao. Tijekom rada na kompleksnim projektima, dolazi do više izazova i prepreka koje je onda i teže riješiti na zadovoljstvo svih strana. Nejasne obveze udaljuju tim od samog cilja cjelokupnog projekta, a zadaća je voditelja projekta da taj izazov eliminira. Nedostatak resursa pretežito se odnosi na dvije skupine – ljudski resursi i tehnički resursi. Navodi se kako manjak navedenih resursa posebice dolazi na naglasak kada se bliži rok isporuke softvera, a projekt ustvari „stoji“ na mjestu ili se izrazito sporo kreće. Između ostalog, neizostavno je navesti izazove za sam softver. Pojavljivanje mnogo „bugova“ (35 %) kao i manjkavosti koda (17.5 %) ističu se kao dva ključna izazova ovog područja:

„Manjkavosti koda događaju se zbog lošeg postavljanja arhitekture i/ili zbog neiskustva programera. Također, u timu postoje iskusniji i manje iskusni programeri koji ne rade na isti način pa to sve utječe na cjelokupni kod.“

„Ovakve rizike suzbijamo redovitim nadzorom tech leada (glavnog developera) koji nadgleda programere te ih savjetuje u radu i ukazuje na potencijalne opasnosti i zajedno s njima vrši reviziju koda.“

Kada je riječ o brojnim „bugovima“ to najčešće podrazumijeva nestabilan i nekonzistentan kod te upućuje na nedovoljno detaljnog testiranja. Kao što je jedan ispitanik naveo: *„Ako je komunikacija među programerima površna, veća je vjerojatnost da će izbiti više bugova.“*. Nadalje, nepotpuni story na kraju svakog sprinta vrijedni su spomena jer taj izazov je označilo 40 % ispitanika. Isto tako druga dva izazova koja imaju značajne postotke odgovora (32.5 % i 30 %) je nedovršavanje testiranja do kraja sprinta kao i provođenje sprintova poput „mini slapova“. Zanimljivo je kako je samo jedan izazov ostao ne označen u potpunosti, a to je izazov da korištenje „storya“ prekida kontinuitet. Može se reći kako je ova lista izazova prikaz uistinu najčešćih izazova s kojima se IT i ICT poduzeća u Hrvatskoj susreću i kako je u jednu ruku zabrinjavajuće s koliko se izazova ustvari poduzeća suočavaju prilikom razvoja softverskih projekta.

No, osim navedenih izazova, ispitanicima se ponudila opcija da navedu i izazove koje je autorica rada previdjela, ukoliko istih ima. Neki od zaprimljenih odgovora su sljedeći:

„Izostanak feedbacka za završene zadatke“

„Neadekvatna i nedovoljna uključenost leadership struktura organizacije u agilnu tranziciju (bez toga uklanjanje prepreka postaje jako teško).“

„Manjak timskog duha, introvertni pojedinci, zahtjevni Vlasnici proizvoda.“

„Česta promjena tehnologije (vanjski utjecaj) utječe na razinu znanja i usporava razvoj novih funkcionalnosti.“

„Neprecizno planiranje sprinteva, kao i potreba za isporuku tijekom sprinta.“

„Pokušavali smo popravit sve odjednom - kad bi naletjeli na izazove i probleme, pokušali bi ih popravit barem pet odjednom, a to nikako nije funkcioniralo. Bolje je fokusirat se na dva do tri problema i njih popravit.“

Pružanje povratne informacije članovima tima omogućuje daljnje napredovanje i u konačnici utječe na adekvatno upravljanje vremenom kao i zadovoljstvo zaposlenika. Također, često je kultura organizacije važna karika u poslovanju, jer opisuje način na koji pojedina organizacija funkcionira. Neadekvatan tim ili vodstvo uvijek će rezultirati neuspjehom projekta.

Iako nisu brojni, ostali navedeni izazovi također su dio poteškoća sa kojima se pojedine organizacije susreću razvijajući softverski projekt. Posebice u ovo moderno doba, izazovi vezani za nove tehnologije su neizbježni i jedina solucija je adaptirati se i pravovremeno educirati o korištenju istih. Neadekvatno planiranje, kao i nepoznanice pri planiranju sprinta ili pak krive procjene programera često utječu na isporuku prototipa i potencijalno nezadovoljstvo klijenta. Nadalje, ispravno postavljanje tima prema izazovima sa kojima se susrećemo, ključ je daljnjeg funkcioniranja. Niti ignoriranje problema niti forsiranje rješavanja istih odjednom, neće dobro rezultirati. Kako bi se spriječilo susretanje sa velikim brojem izazova važno je znati da je pravilno upravljanje softverskim projektom posao cijelog tima, a ne samo jedne osobe. Stoga, ukoliko i dođe do problema, njih treba analizirati i pristupiti im na odgovarajući način kako bi se isti riješili brzo i efikasno.

4.3. Ograničenja istraživanja i preporuke za buduća istraživanja

Istraživanje o izazovima sa kojima se susreću poduzeća prilikom korištenja Scrum okvira za agilno upravljanje projektima je provedeno putem ankete upućene u 28 hrvatskih IT i ICT firmi. Iako je prednost ankete anonimnost odgovora, uvijek postoji određeni stupanj nesigurnosti oko toga koliko je ispitanik istinit ili koliko se posvetio pisanju svojih odgovora. Drugim riječima, postoji i dalje mogućnost da su se ispitanici suzdržavali oko detalja i opširnosti prilikom odgovaranja i davali šture odgovore što zbog mišljenja da ne smiju dijeliti određene informacije o poduzeću, a što zbog manjka volje za pisanjem i obrazlaganjem odgovora. Nadalje, jedan od ograničenja je i subjektivnost, jer su izazovi istraženi većim dijelim iz viđenje jedne osobe (na primjer projektni menadžer u poduzeću, dok bi izazovi bili reprezentativniji da su ih birali i objašnjavali recimo i Scrum Master i projektni menadžer i developer).

Preporuka za buduće istraživanje je provesti i dubinski intervju jer bi sa izravnom komunikacijom sa ispitanicima bili dobiveni detaljniji i opširniji odgovori naspram odgovora iz ankete jer ljudi ne vole kada moraju previše pisati. Nadalje, preporuka je i ispitati izazove po određenoj skupini, tj. fokusirati se samo na jedan segment Scrum okvira i u njemu istražiti izazove (na primjer izazovi po ulogama u timu, po veličini), a ne općenito za Scrum kako bi se dobio bolji uvid u problematiku pojedinog područja. Osim toga, preporuka je i proširiti istraživanje i istražiti izazove drugih metodologija van agilne domene u IT i ICT sektoru, poput Kanbana ili Leana.

5. ZAKLJUČAK

Agilne metode upravljanja projektima za razvoj softvera razvile su se kako bi kupcima pružile više zadovoljstva, smanjile troškove, skratile razvojni životni ciklus proizvoda, smanjile broj grešaka te u konačnici kako bi se prilagodile promjenjivim poslovnim zahtjevima tijekom razvojnog procesa softvera. S obzirom na dinamičnost u kojem posluju IT i ICT poduzeća Scrum okvir za agilno upravljanje softverskim projektima pokazao se kao pravo rješenje za takvu industriju. Scrum ustvari predstavlja procesni okvir za rješavanje složenih problema ili situacije i isporuku proizvoda s najvećom mogućom vrijednošću.

Empirijskim istraživanjem utvrđeno je kako velik broj čimbenika utječe na razvoj softvera prilikom implementiranja Scrum okvira. Istraživanje prolazi kroz 39 glavnih izazova ispitanih unutar 28 hrvatskih IT i ICT poduzeća. Iako je Scrum često okarakteriziran svojim prednostima, ovo istraživanje pokazalo je kako se organizacije na dnevnoj bazi susreću sa značajnim brojem izazova. Izrazito je važno takvu problematiku promatrati iz očiju relevantnih osoba unutar projektnog tima. Tako su projektni menadžeri, Scrum Masteri, Vlasnici proizvoda, developeri i brojni drugi, kao ključne, tj. primarne osobe svakog projekta, ovu anketu obogatile svojim detaljnim odgovorima. Rezultati su pokazali kako je manjak znanja o Scrum domeni čest razlog brojnih drugih problema, a neadekvatna edukacija logičan je uzrok tomu. Također, nekvalitetno upravljanje vremenom i budžetom, te mnogo „bugova“ često čini prototip neisporučivim. Istodobno, rezultatima se utvrdilo kako posebno treba pripaziti na odnos s klijentima jer njihova neodlučnost te manjak spoznaje o željenim značajkama u softveru, nosi specifičan set izazova poput konstantnih izmjena zahtjeva. Takvi izazovi vuku brojne druge probleme sa sobom i čine upravljanje projektima izrazito teškim. Ovim istraživanjem uočeno je kako se teorijski dio doista podudara sa poslovnom praksom.

Usprkos brojnim izazovima sa kojima se organizacije susreću, softverski projekti u hrvatskom IT i ICT sektoru uspješno se realiziraju. No, bolja organizacijska kultura, komunikacija te bolji odnosi spram samog procesa upravljanja projektima olakšati će rad i smanjiti izazove prilikom upravljanja projektima primjenom Scrum okvira.

POPIS IZVORA

1. Abrahamsson, P., Salo, O., Ronkainen, J. i Warsta, J. (2002.), *Agile software development methods*, Espoo: VTT Otamedia Oy
2. Akbar, R. i Safdar, S. (2015.), A short review of Global Software Development (GSD) and latest software development trends, u: I*nternational Conference on Computer, Communications, and Control Technology (I4CT)* (str. 314 - 317.), Kuching: IEE
3. Ashmore, S. i Runyan, K. (2014.), *Introduction to Agile Methods*, Boston: Addison-Wesley Professional
4. Atlassian Agile Coach (b. d.), What is a scrum master?, preuzeto 1. kolovoza s https://www.atlassian.com/agile/scrum/scrum-master
5. Batra, D., Sin, T. i Tseng, S. (2006.), Modified agile practices for outsourced software projects, u: Rodríguez Abitia G. i Ania B. I. (ur.), *Proceedings of the twelfth Americas Conference on Information Systems*, (str. 3872-3880.), Akapulko: Association for Information Systems
6. Beck, K. (2000.), *Extreme Programming Explained: Embrace Change*, 2. izd., Boston: Addison-Wesley Professional
7. Bentley, P. (2005.), Investigations Into Graceful Degradation of Evolutionary Developmental Software, *Nat Comput*, *4*, 417–437. https://doi.org/10.1007/s11047-005-3666-7
8. Cao, L., Kannan, M., Xu, P. i Ramesh, B. (2017.), A framework for adapting agile development methodologies, *European Journal of Information Systems Volume, 18*(4) 332-343.
9. Cervone, H. F. (2011.), Understanding agile project management methods using Scrum. *OCLC Systems Serv., 27*, 18-22., doi: https://doi.org/10.1108/10650751111106528
10. Childs, S. i McLeod, J. (2013.), Tackling the wicked problem of ERM: using the Cynefin framework as a lens, *Records Management Journal*, *23*(3), 191–227. doi:10.1108/rmj-07-2013-0016
11. Cho, J. (2007.), Globalization and global software development, *Journal of IACIS-Issues in Information Systems*, *8*(2), 287-290.
12. Cho, J (2010.), An Exploratory Study on Issues and Challenges of Agile Software Development with Scrum, *All Graduate Theses and Dissertations*, *9*(2), 188-195.
13. Cockburn, A. i Highsmith, J. (2001.), Agile software development: the business of innovation, *Computer, 34*(9), 120-127., doi:10.1109/2.947100
14. Cohen, C. F., Birkin, S. J., Garfield, M. J. i Webb, H. W. (2004.), Managing conflict in software testing, *Communications of the ACM*, *47*(1), 76–81. doi:10.1145/962081.962083
15. Čubranić, D., Kaluža, M. i Novak J. (2013.), Standardne metode u funkciji razvoja softvera u Republici Hrvatskoj, *Zbornik Veleučilišta u Rijeci*, *1*(1), 239-256., Preuzeto s https://hrcak.srce.hr/103336
16. Dey, P. P., Khan, M., Amin, M., Sinha, B. R. i Badkoobehi H. (2016.), Software Project Management Challenges, *International Review of Research in Emerging Markets and the Global Economy (IRREM) - An Online International Research Journal, 2*(1), 787-796. http://globalbizresearch.org/files/kf528_irrem_pradip-peter-dey_muzibul-khan_mohammad-amin_bhaskar-raj-sinha_hassan-badkoobehi-385113.pdf

17. Despa, M. L. (2014.), Comparative study on software development methodologies, *Database Systems Journal*, *5*(3), 37-56.
18. Dybå, T., Dingsøyr, T. i Moe, N. B. (2014.), Agile Project Management, u: Ruhe, G., i Wohlin, C. (ur.), *Software Project Management in a Changing World* (str. 277-300.), Berlin: Springer
19. Erickson, J. i Ranganathan, C. (2006.), Project Management Capabilities: Key to Application Development Offshore Outsourcing u: *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06), 8*, 1-10.
20. Faiza, A., Shabib, A., Usman, W. i Syed Shah, M. (2017.), Agile Software Development Models TDD, FDD, DSDM, and Crystal Methods: A Survey, *International Journal of Multidisciplinary Sciences and Engineering*, *8*(2), 1-10.
21. Fernandez, D. J. i Fernandez, J. D. (2008.), Agile Project Management —Agilism versus Traditional Approaches, *Journal of Computer Information Systems, 49*(2), 10-17. https://doi.org/10.1080/08874417.2009.11646044
22. Fierro, D., Putino, S. i Tirone L. (2018.), The Cynefin Framework and Technical Competencies: a New Guideline to Act in the Complexity, *INCOSE Proceedings, 28*(1), 532-552. 10.1002/j.2334-5837.2018.00498.x
23. French, S. (2013.), Cynefin, statistics and decision analysis, *Journal of the Operational Research Society*, *64*(4), 547–561. https://doi.org/10.1057/jors.2012.23
24. Froggyads (b. d.), Scrum Methodology: Project Management Guide, preuzeto 1. kolovoza s https://froggyads.com/blog/scrum-methodology-project-management-guide/
25. Fustik, V. (2017.), The advantages of agile methodologies applied in the Ict development projects, *International Journal on Information Technologies & Security*, *9*(4), 51-62.
26. GeeksforGeeks (2020., 26. kolovoz), Different types of risks in Software Project Development, preuzeto 6. kolovoza s https://www.geeksforgeeks.org/different-types-of-risks-in-software-project-development/
27. GeeksforGeeks (2018., 02. kolovoz), Software Engineering, Software Project Management Complexities, preuzeto 6. kolovoza s https://www.geeksforgeeks.org/software-engineering-software-project-management-complexities/
28. Gonçalves, L. (2018.), Scrum, *Controlling & Management Review*, *62*(4), 40–42. doi:10.1007/s12176-018-0020-3
29. Han, W. M i Huang, S. J. (2007.), An empirical analysis of risk components and performance on software projects, *Journal of Systems and Software, 80*(1), 42–50.
30. He, W., Zhang, J. i Li, W. (2020.), Information Technology Solutions, Challenges, and Suggestions for Tackling the COVID-19 Pandemic, *International Journal of Information Management*, 57, 1-22. https://doi.org/10.1016/j.ijinfomgt.2020.102287
31. Herbsleb, J. i Moitra, D. (2001.), Global software development, *IEEE software*, *18*(2), 16-20. http://dx.doi.org/10.1109/52.914732
32. Hetzel, W. C. (1984.), *The Complete Guide to Software Testing*, Wellesley: Q.E.D. Information Sciences
33. Highsmith, J. (2009.), *Agile Project Management: Creating Innovative Products*, 2. izd., Boston: Addison-Wesley Professional
34. Highsmith, J. (2013.) *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, Boston: Addison-Wesley Professional

35. Hunt, J. (2006.), *Agile Software Construction*, London: Springer
36. Huynh, Q. T. i Nguyen, N. T. (2020.), Probabilistic Method for Managing Common Risks in Software Project Scheduling Based on Program Evaluation Review Technique, *International Journal of Information Technology Project Management*, *11*(3), 77–94. doi:10.4018/ijitpm.2020070105
37. Juhnke, K., Tichy, M. i Houdek, F. (2020.), Challenges concerning test case specifications in automotive software testing: assessment of frequency and criticality, *Software Quality Journal*, *29*, 39-100. doi:10.1007/s11219-020-09523-0
38. Jurison, J. (1999.), Software Project Management: The Manager's View, *Communications of the Association for Information Systems*, *2*(17), 1-57. https://doi.org/10.17705/1CAIS.00217
39. Keith, C. (2010), *Agile game development with Scrum*, Crawfordsville: Addison-Wesley
40. Khan, A. I., Ali Khan, U. i Qurashi, R. J. (2011.), A Comprehensive Study of Commonly Practiced Heavy and Light Weight Software Methodologies, *IJCSI International Journal of Computer Science Issues*, *8*(4), 441-450
41. Kilibarda, G. D., Šobajić, V. M., Berić, I. M. i Jovanović, P. M. (2016.), *Upravljanje softverskim projektima* [e-publikacija], preuzeto s https://scindeks-clanci.ceon.rs/data/pdf/0040-2176/2016/0040-21761601145K.pdf
42. Klonek, F. E., Kanse, L., Wee, S., Runneboom, C. i Parker, S. K. (2021.), Did the COVID-19 Lock-Down Make Us Better at Working in Virtual Teams? *Small Group Research*, *52*(2), 1-22. doi:10.1177/10464964211008991
43. Layton, M. C. (2012.), *Agile Project Management For Dummies*, 1.izd., Hoboken: John Wiley & Sons
44. Learntek (2019., 25. siječanj), SDLC (Software Development Life Cycle), preuzeto 8. rujna s https://www.learntek.org/blog/sdlc-phases/
45. Leau, Y., Loo, W. K., Tham, W. Y. i Tan, S.F. (2012.), Software Development Life Cycle AGILE vs Traditional Approaches u: *International Conference on Information and Network Technology* (str. 162-167.), Singapur: IACSIT Press
46. Liović, D. (2016.), Outsourcing – rizična ušteda? u: Stanišić, S., Stanišić, N. i Petrović, Z. (ur.), *Singidunum University International Scientific Conference, risks in contemporary business*, (str. 223-230.), Beograd: Sveučilište Singidunum
47. Luckey, T. i Phillips, J. (2006.), *Software Project Management For Dummies*, Hoboken: Wiley Publishing
48. Mahalakshmi, M. i Sundararajan, M. (2013.), Traditional SDLC Vs Scrum Methodology – A Comparative Study, *International Journal of Emerging Technology and Advanced Engineering*, Journal, *3*(6), 192-196.
49. Maximilien, M. E. i Campos, P. (2012.), Facts, trends and challenges in modern software development, *International Jounral of Agile and Extreme Software Development*, *1*(1), 1-5.
50. Miller, C., Rodeghero, P., Storey, M. A., Ford, D. i Zimmermann, T. (2021). How Was Your Weekend? - Software Development Teams Working From Home During COVID-19. u: O'Conner, L. (ur.), *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)* (str. 624-636.), Piscataway: IEEE
51. MindTools (b. d.), The Cynefin Framework - Using the Most Appropriate Problem-Solving Process, preuzeto 8. rujna 2021. s https://www.mindtools.com/pages/article/cynefin-framework.htm

52. Myers, G. J. (1979.), *Art of software Testing*, New York: John Wiley & Sons
53. Ncube (2021., 19. srpanj), Software Development Life Cycle: A Guide to Phases and Models, preuzeto 5. kolovoza s https://ncube.com/blog/software-development-life-cycle-guide
54. O'Connor, R. V. i Lepmets, M. (2015.), Exploring the use of the cynefin framework to inform software development approach decisions u: Pfahl, D. (ur.), *Proceedings of the 2015 International Conference on Software and System Process*, (97-101.), New York: Association for Computing Machinery
55. Palacio, M., (2021.), *Scrum Master*, preuzeto s https://scrummanager.net/files/scrum_master_en.pdf
56. Pdfprof (b. d.), Télécharger application development life cycle, preuzeto 9. rujna s https://www.pdfprof.com/PDF_Image.php?idt=37475&t=28
57. Pirozzi, M. (2018.), The Stakeholder Management Perspective to increase the Success Rate of Complex Projects, *PM World Journal*, *7*(1), 1-12.
58. PMI (2017.), *Agile Practise Guide*, Pennsylvania: Project Menagement Institute
59. Puik, E., i Ceglarek, D. (2015.), The Quality of a Design will not Exceed the Knowledge of its Designer; an Analysis Based on Axiomatic Information and the Cynefin Framework, *Procedia CIRP, 34*, 19–24. doi:10.1016/j.procir.2015.07.040
60. Pyhäjärvi, M. i Rautiainen, K. (2004.), Integrating Testing and Implementation into Development, *Engineering Management Journal*, *16*(1), 33–39. https://doi.org/10.1080/10429247.2004.11415236
61. Rao, M. T. (2004.), Key Issues for Global it Sourcing: Country and Individual Factors, *Information Systems Management*, *21*(3), 16–21.
62. Redmond-pyle, D. (1996.), Software development methods and tools: some trends and issues, *Software Engineering Journal*, *11*(2), 99. doi:10.1049/sej.1996.0013
63. Rehman, A. i Hussain, R. (2007.), Software Project Management Methodologies/Frameworks Dynamics - A Comparative Approach u: 2007. *International Conference on Information and Emerging Technologies* (str. 1-5.) Karachi: IEE
64. Rigby, D. K., Sutherland J. i Takeuchi H. (2016.), Embracing Agile: How to Master the Process That's Transforming Management, *Harvard Business Review*, *94*(5), 40–50.
65. Robson, S. (2013.), *Agile SAP: Introducing Flexibility, Transparency and Speed to SAP Implementations*, Cambridge: IT Governance Publishing
66. Rubin, K. S. (2012.), *Essential Scrum: A Practical Guide to the Most Popular Agile Process*, 1.izd., Boston: Addison-Wesley Professional
67. SamSolutions (b. d.), Ten Software Development Trends for 2020–2021, preuzeto 3. kolovoza s https://www.sam-solutions.com/blog/software-development-trends/
68. Schach, S. R. (1996.), Testing: principles and practice, *ACM Computing Surveys*, *28*(1), 277–279. doi:10.1145/234313.234422
69. Schneckenberg, D., Benitez, J., Klos, C., Velamuri, V. K. i Spieth, P. (2021.), Value creation and appropriation of software vendors: A digital innovation model for cloud computing, *Information & Management, 58*(4), 1-14. doi:10.1016/j.im.2021.103463
70. Schwaber, K. i Sutherland, J. (2017.*), The Scrum Guide, The Definitive Guide to Scrum: The Rules of the Game* [e-publikacija], preuzeto s https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf
71. Schwaber, K. i Beedle, M. (2002.), *Agile software development with Scrum*, Upper Saddle River: Prentice Hall

72. Schwalbe, K. (2016.), *Information Technology Project Management*, 8. izd., Boston: Cengage learning
73. SCRUMstudy (b. d.), What is Crystal?, preuzeto 11. srpnja 2021. s http://blog.scrumstudy.com/what-is-crystal/
74. Scrum.org (b. d.), What is a Product Owner? Learn About the Role of the Product Owner, preuzeto 28. srpnja 2021. s https://www.scrum.org/resources/what-is-a-product-owner
75. Słoniec, J. (2015.), Use of Cloud Computing in Project Management, *Applied Mechanics and Materials*, *791*, 49–55. https://doi.org/10.4028/www.scientific.net/AMM.791.49
76. Srivastava, A., Bhardwaj, S. i Saraswat, S. (2017.), SCRUM model for agile methodology u: Parma Nand A., Abhishek S., Vishnu S., Manjeet, S. i Kesav G. (ur.), *2017 International Conference on Computing, Communication and Automation (ICCCA)* (str. 864-869.), Piscataway: IEEE
77. Stare, A. (2013.), Agile project management – a future approach to the management of projects*?*, *Dynamic Relationships Management Journal*, *2*(1), 43-53. http://dx.doi.org/10.17708/DRMJ.2013.v02n01a04
78. Stellman, A. i Greene, J. (2014.), *Learning Agile: Understanding Scrum, XP, Lean, and Kanban*, Sebastopol: O'Reilly Media
79. Sudhakar, G. P. (2010.), *Elements of Software Project Management*, 1. izd., New Delhi: PHI Learning Pvt. Ltd.
80. Sverrisdottir, H. S., Ingason, H. T. i Jonasson, H. I. (2014.), The Role of the Product Owner in Scrum-comparison between Theory and Practices, *Procedia - Social and Behavioral Sciences*, *119*, 257–267. doi:10.1016/j.sbspro.2014.03.030
81. Takeuchi, H. i Nonaka, I. (1986.), The New New Product Development Game, *Harvard Business Review*, *64*(1), 152.
82. Talby, D., Keren, A., Hazzan, O. i Dubinsky, Y. (2006.), Agile software testing in a large-scale project, *IEEE Software*, *23*(4), 30-37.
83. TXM (b. d.), Making sense of your problems with the cynefin framework, preuzeto 1. rujna s https://txm.com/making-sense-problems-cynefin-framework/
84. Tyran, C. K. (2006.), A software inspection exercise for the systems analysis and design course, *Journal of Information Systems Education*, *17*(3), 341-351.
85. Upwork (2020., 19. listopad), What Is the Software Development Life Cycle (SDLC) and Which Method Is the Best?, preuzeto 5. kolovoza s https://www.upwork.com/resources/what-is-software-development-life-cycle?utm_source=google&utm_campaign=SEM_GGL_INTL_NonBrand_Marketplace_DSA&utm_medium=cpc&utm_content=113089129402&utm_term=&campaignid=11384804789&matchtype=b&device=c&gclid=CjwKCAjwmK6IBhBqEiwAocMc8kjmKAGWSVnpzKh2QPpiun8Ec3-eMII5qzR6DBgftLpyjY4zHAfZmRoCC0oQAvD_BwE
86. Van Beurden, E. K., Kia, A. M., Zask, A., Dietrich, U. i Rose, L. (2011.), Making sense in a complex landscape: how the Cynefin Framework from Complex Adaptive Systems Theory can inform health promotion practice, *Health Promotion International*, *28*(1), 73–83. doi:10.1093/heapro/dar089
87. Vanzant Stern, T. (2017.), *Lean and Agile Project Management*, New York: Productivity Press

88. Visual Paradigm (b. d.), What are Scrum Artifacts?, preuzeto 1. kolovoza 2021. s https://www.visual-paradigm.com/scrum/what-are-scrum-artifacts/
89. Wrike (b. d.), What Is Software Project Management?, preuzeto 2. kolovoza 2021. s https://www.wrike.com/project-management-guide/faq/what-is-software-project-management/
90. Wysocki R. K. (2014.), *Effective Software Project Management: Traditional, Agile, Extreme*,7. izd., Indianopolis: John Wiley & Sons
91. Yung, C. i Lin, Y. T. (2015.), Implementing TOAST, a Tool for Agile Software Project Management in Cloud Computing Environments, *Journal of Software*, *10*(11), 1310-1318. doi: 10.17706/jsw.10.11.1310-1318
92. Zhang, X., Hu, T. Hua D. i Li X. (2010.), Software Development Methodologies, Trends and Implications: A Testing Centric View, *Information Technology Journal*, *9*, 1747-1753. 10.3923/itj.2010.1747.1753

POPIS SLIKA

POPIS TABLICA

ŽIVOTOPIS AUTORICE

Ana Radoš

Rebro 50, 10360 Sesvete

Mobitel: 0915022661

Email: ana-rados@hotmail.com

RADNO ISKUSTVO

Administrativna radnica 03/2020 – danas

Radoš trgovina d.o.o.

• Izvedena i vođena produkcija računovodstvene aktivnosti u cijeloj organizaciji

• Svakodnevno, tjedno i mjesečno vršeno usklađivanje računa

• Radila u odijelu za ljudske resurse - proces zapošljavanja

• Radila u Adobe Photoshopu u svrhe marketinga

• Odgovarala na telefonske pozive

Voditeljica objekta 05/2019 – 03/2020

Studio apartmani Radoš

• Upravljanje sa 6 studio apartmana koji su dio obiteljskog poduzeća

• Prijavljivanje i odjavljivanje gostiju

• Odgovorna za organizaciju usluga čišćenja

Demonstratorica na kolegiju Marketing 02/2019 – 09/2019

Ekonomski Fakultet u Zagrebu

• Pomaganje i podučavanje učenika da savladaju i bolje razumiju materijale i gradivo

Edukacija

Magisterij: Integrirani diplomski i preddiplomski sveučilišni studij, smjer Menadžment

Ekonomski fakultet Zagreb, 2016. godina - danas

Jezične vještine

Hrvatski – Materinski jezik

Engleski B2 – u govoru, pismu i slušanju

Španjolski B1 – u govoru, pismu i slušanju

Slovenski A1 – u govoru, pismu i slušanju

Njemački A1 - u govoru, pismu i slušanju

Digitalne vještine i ostale vještine

Internet / MS Office (Word Excel PowerPoint) / Rad na računalu / Microsoft Word / Microsoft Excel / Komunikacijski programi (Skype, Zoom, TeamViewer) /Timski rad / Informacije i komunikacija / Microsoft NAV Dynamics / Prilagodljivost / Pregovaračke vještine / Digitalni marketing / Osnovno znanje Adobe Photoshopa