UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

MAGISTRSKO DELO

DIGITALIZACIJA PROCESA TESTIRANJA PROGRAMSKIH REŠITEV

Ljubljana, 14. januar 2019 ŠPELA MARINČIČ

IZJAVA O AVTORSTVU

Podpisana Špela Marinčič, študentka Ekonomske fakultete Univerze v Ljubljani, avtorica predloženega dela z naslovom Digitalizacija procesa testiranja programskih rešitev, pripravljenega v sodelovanju s svetovalko red. prof. dr. Mojco Indihar Štemberger

I Z J A V L J A M

1. da sem predloženo delo pripravila samostojno;

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

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

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

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

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

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

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

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

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

V Ljubljani, dne _____ Podpis študentke: _____

KAZALO

KAZALO TABEL

KAZALO SLIK

KAZALO PRILOG

SEZNAM KRATIC

ang. – angleško

AS–IS model – (ang. AS–IS model); Obstoječi model poslovnega procesa

BPEL – (ang. Business Process Execution Language); Jezik za modeliranje poslovnih procesov

BPMN – (ang. Business Process Management Notation); Grafični prikaz procesov v modelu poslovnega procesa

CMMI – (ang. Capability Maturity Model Integration); Zrelostni model za vrednotenje kakovosti razvoja programske opreme

CRM – (ang. Customer Relationship Management); Sistem za upravljanje odnosa s strankami
DMS – (ang. Document Management System); Dokumentni sistem
ERP – (ang. Enterprise resource planning); Integrirani poslovno – informacijski sistem
EUR – (ang. euro); Evro
MPP – (ang. Business Process Management); Management poslovnih procesov
orodja MPP – (ang. Business Process Management Tools); Orodja za podporo managementu poslovnih procesov
SDLC – (ang. Software development life cycle for software development); Življenjski cikel razvoja programskih rešitev
TO–BE model – (ang. TO–BE model); Model prenovljenega poslovnega procesa
UML – (ang. Unified Modeling Language); Jezik za modeliranje poslovnih procesov

UVOD

Dandanes se podjetja na trgu srečujejo z veliko konkurenco, z vedno večjimi zahtevami strank in s hitro spreminjajočim se poslovnim okoljem. Za uspešno podjetje ni dovolj, da na trgu ponudi izdelek ali storitev, ki mu omogoča rast prihodkov in dobička, temveč je zelo pomembno, da je njegovo poslovanje uspešno. O uspešnosti poslovanja podjetja govorimo takrat, ko posluje z najnižjimi možnimi stroški, poslovni procesi se odvijajo v najkrajšem možnem času, pri čemer je njihova izvedba kakovostna, podjetje pa pri izvedbi procesov uporablja sodobno informacijsko tehnologijo, ki jo prilagaja glede na potrebe poslovanja (Kovačič, Jaklič, Indihar Štemberger & Groznik, 2004).

Podjetje izvaja in prenavlja, nadgrajuje in dopolnjuje poslovne procese z namenom, da uresniči različne cilje, npr. poveča dobiček, se razširi na nove trge, izboljša pretok informacij znotraj podjetja in med podjetji, s katerimi sodeluje, izboljša komunikacijo med stranko in podjetjem, izvaja hitrejšo dostavo blaga stranki, razvije izdelek ali storitev, ki bo podjetju pripomogla k večji prepoznavnosti, dobičkonosnosti itd. Določenih ciljev podjetje ne more doseči z obstoječimi procesi, zato jih mora nadgraditi, prenoviti, avtomatizirati in optimizirati (Dumas, La Rosa, Mendling & Reijers, 2013).

Vodilni zaposleni v podjetjih se čedalje bolj zavedajo, kako so za uspešno poslovanje pomembni dobro zastavljeni poslovni procesi, a jim kljub temu v nekaterih podjetjih tako v Sloveniji kot v tujini še vedno namenijo premalo pozornosti. Načrti in ideje, kako izboljšati poslovanje podjetja prek prenove poslovnih procesov, žal pogosto še vedno ostajajo samo na papirju ali pa v glavah zaposlenih (Križevnik & Branko Jurič, 2009). Velika težava nastane, ko zaposleni ne morejo več obvladovati poslovnih procesov, ker jih premalo poznajo, ko nihče od vključenih v proces ne ve, kako poteka celotni proces od začetka do konca, ko procesi niso prilagojeni informacijski tehnologiji in obratno, ko se aktivnosti v procesih podvajajo, pojavljajo se ponavljajoča se opravila, s katerimi se izgublja čas in se povečujejo stroški itd. (Desel, Oberweis & Van Der Alast, 2000, str. 6). Takrat je smiselno, da podjetja začnejo ukrepati, bodisi s prenovo določenih procesov ali pa z radikalno prenovo poslovanja.

V magistrskem delu sem se osredotočila na pomembno vlogo uporabe napredne programske opreme, natančneje uporabe orodij za podporo managementu poslovnih procesov (ang. Business Process Management Tools, v nadaljevanju orodja MPP), ki omogočajo, da z njihovo pomočjo podjetja poslovne procese optimizirajo in avtomatizirajo ter s tem dosežejo digitalizacijo poslovanja. Današnja orodja MPP omogočajo podjetjem učinkovito in uspešno obvladovati poslovne procese in prilagajati korake v procesih glede na poznavanje vsebine procesov, podatkov ter poslovnih dogodkov (Moore in drugi, 2017, str. 12). Potrebno je poudariti, da podjetje ne bo doseglo željene stopnje digitalizacije poslovanja s pomočjo informacijske tehnologije, če ne bo pred tem temeljito razdelalo poslovnih procesov, jih popisalo, poenotilo, spremenilo itd. Lahko se celo zgodi, da pri nepravilni integraciji poslovnih procesov v programsko opremo podjetje občuti negativne

posledice v poslovanju (Kovačič, Jaklič, Indihar Štemberger & Groznik, 2004).

Obvladovanje poslovnih procesov v podjetju je v zadnjih letih predmet preučevanja tako poslovnih ved kot tudi ved s področja informacijske tehnologije. Poslovne vede preučujejo poslovne procese v podjetjih z vidika, kako izboljšati poslovanje podjetja. V ta namen iščejo odgovore na naslednja vprašanja: kako povečati prepoznavnost blagovne znamke, kakšna mora biti strategija podjetja, da se bodo zmanjšali stroški poslovanja, na kakšen način povečati zadovoljstvo strank, ali mora podjetje razviti kakšno novo storitev ali izdelek, da poveča svojo konkurenčno prednost na trgu itd. (Weske, 2007, str. 2).

Vede s področja informacijske tehnologije pa po drugi strani preučujejo strukturne lastnosti poslovnih procesov. Poslovni analitiki, ki se ukvarjajo s preučevanjem poslovnih procesov in njim prilagojene informacijske tehnologije, so zadolženi, da poslovne procese prenesejo iz podjetniškega poslovnega okolja v računalniški jezik. To pomeni, da procese v začetni fazi popišejo in jih izrišejo v obliki diagramov, v orodjih MPP, ki omogočajo modeliranje, analizo, optimizacijo in digitalizacijo procesov. Pomembno je, da so procesi podrobno razdelani, saj le tako lahko analitiki opazijo ozka grla in prepoznajo potencialne možnosti digitalizacije procesov s pomočjo programske opreme (Weske, 2007, str. 3). V prihodnjih letih bo eden izmed najbolj zaželenih poklicev na področju upravljanja s poslovnimi procesi prav poslovni analitik oziroma poslovni tehnolog, ki bo tako v manjših kot v večjih podjetjih deloval predvsem na področju digitalizacije poslovanja, v sodelovanju s strokovnjaki s področja razvoja programskih rešitev. Poslovni analitik ima tako tehnična kot vsebinska znanja s področja digitalizacije poslovanja in poznavanja poslovnih procesov ter tako pomaga podjetjem, da nenehno izboljšujejo in nadgrajujejo poslovne procese (Pratt & White, 2018).

Procesi so in bodo čedalje bolj integrirani v aplikacije. S pomočjo programske opreme se lahko izvede celoten proces ali pa samo del procesa, preostanek aktivnosti v procesu pa še vedno izvajajo zaposleni. Aplikacije, ki jih uporabljajo podjetja, niso pomembne samo za avtomatizacijo procesov, ampak tudi za boljši pregled nad izvajanjem celotnega procesa (npr. podatki so shranjeni v podatkovnih skladiščih tako, da zaposleni iz različnih oddelkov lahko do njih dostopajo v vsakem trenutku, jih zato hitreje obdelajo, posledično pa je izmenjava informacij med zaposlenimi znotraj podjetja in z zunanjim poslovnim okoljem hitrejša, učinkovitejša ter pravilnejša) (Weske, 2007).

Podjetje lahko s pomočjo dobro organiziranih poslovnih procesov doseže konkurenčno prednost na različnih področjih, kot npr. prodaja, nabava, oskrbna veriga, proizvodnja itd. Zara je primer podjetja, ki dosega konkurenčno prednost ne samo pri prodaji modnih oblačil in dodatkov, ampak tudi na področju dobro razdelanih logističnih procesov vzdolž oskrbne verige. Pomembno vlogo pri učinkoviti in uspešni oskrbni verigi ima napredna informacijska tehnologija. Zara je vertikalno integrirano podjetje. Design oblačil in obutve, večino proizvodnje produktov, skladiščenje, distribucijo ter logistiko opravi podjetje samo. Ima zelo jasno razdelane procese in stroga pravila pri naročanju ter dostavi v prodajalne po

Evropi in drugod po svetu. Poudariti je potrebno, da brez napredne tehnologije v proizvodnji in brez informacijske tehnologije, ki omogoča hitro ter učinkovito beleženje zalog, prodanih artiklov v trgovinah po svetu itd. Zara zagotovo ne bi imela tako uspešne oskrbne verige (Simchi–Levi, Kaminsky, Simchi–Levi & Shankar, 2007). Je primer podjetja, ki se močno zavzema za digitalizacijo poslovanja.

V začetku devetdesetih let se je začelo razvijati pomembno področje, ki se ukvarja z upravljanjem poslovnih procesov v podjetju, in sicer management poslovnih procesov (ang. Business Process Management – BPM, v nadaljevanju MPP) (Lusk, Paley & Spanyi, 2005). Bistvo MPP je, da nenehno išče izboljšave in nadgradnje poslovanja tudi v povezavi z informacijsko tehnologijo.

Prenova poslovanja podjetja se lahko izvede interno ali pa se na trgu poišče podjetje, katerega primarna dejavnost je razvoj programskih rešitev, s katerimi se digitalizira poslovanje in nudi svetovanje glede izvedbe celovite prenove poslovanja. Prenova poteka od analize obstoječega stanja poslovnih procesov do vzpostavitve nove informacijske tehnologije, ki omogoča digitalizacijo poslovnih procesov.

Digitalna preobrazba opisuje spremembe, ki so povezane z uporabo napredne informacijske tehnologije praktično na vseh področjih človeške družbe. Konkretno digitalizacija poslovanja izboljša učinkovitost, uspešnost in agilnost poslovanja, omogoča povečanje inovativnosti v razvoju produktov, zmanjša število rutinskih opravil, omogoča elektronsko hrambo dokumentov ter pomaga podjetjem, da izboljšajo odnos do strank (Moore in drugi, 2017).

Po podatkih tujih podjetij, ki se ukvarjajo z digitalizacijo poslovanja in s svetovanjem pri prenovi poslovanja, naj bi podjetja, ki se odločijo za celovito prenovo poslovanja, imela vidne rezultate v poslovanju že v roku enega do treh mesecev po vpeljavi novega načina izvajanja poslovnih procesov. Njihovi stroški poslovanja naj bi se znižali do 40 odstotkov, produktivnost izvedenih procesov naj bi se povečala med 60 in 80 odstotkov, boljši pa naj bi bil tudi odnos med podjetjem in njihovimi strankami (Dallas & Thandar Wynn, 2014).

V magistrskem delu sem v podjetju, katerega primarna dejavnost je razvoj programskih rešitev za finančne institucije, popisala temeljni proces – testiranje programskih rešitev in ga skušala izboljšati s pomočjo orodja MPP Bizagi Modeler in Bizagi Studio. Testiranje programskih rešitev je proces, ki nastopi za procesom razvoja programske rešitve. Proces testiranja je v celotnem življenjskem ciklu razvoja programske rešitve za podjetje zelo pomemben. Zahteva namreč čas, znanje testerja in lahko za naročnika predstavlja precejšno težavo, predvsem s časovnega in stroškovnega vidika, če med procesom testiranja pride do odkritja nepričakovanih napak, ki podaljšajo rok predaje programske rešitve naročniku ter povečajo stroške razvoja in testiranja za podjetje, ki razvija programske rešitve. Zato je zelo pomembno, da imajo v podjetju proces testiranja dobro razdelan. Zaposleni, ki so vključeni v proces testiranja, morajo dobro poznati celotni

proces, ne samo testiranja programske rešitve, ampak tudi razvoja celotne programske rešitve. Le tako bodo namreč hitreje in učinkoviteje testirali programske rešitve in stranki ponudili produkt po konkurenčni ceni v najkrajšem možnem času.

Naročniki programske rešitve želijo od podjetij, ki nudijo produkte za njihovo poslovanje, razvoj programske rešitve v najkrajšem možnem času in po najnižji mogoči ceni. Največkrat se težave pojavijo po predaji programske rešitve naročniku. Po zaključku razvoja in testiranja programske rešitve se namreč lahko pojavijo nepričakovane napake, ki jih mora ponudnik programskih rešitev nemudoma odpraviti.

Na kakšen način se lahko doseže, da se v procesu testiranja najde čim več napak in da stranka dobi programsko rešitev v najkrajšem možnem času? To je mogoče doseči le v primeru dobro razdelanega in jasnega procesa testiranja programskih rešitev.

Osrednji namen magistrskega dela je izboljšanje temeljnega poslovnega procesa – testiranje programskih rešitev v podjetju, ki se ukvarja z razvojem programskih rešitev za finančne institucije.

Cilji magistrskega dela so:

- Opredelitev ključnih pojmov s področja managementa poslovnih procesov in digitalizacije poslovanja.
- Preučiti Porterjevo verigo vrednosti in Leavittov diamant v splošnem in ju aplicirati na izbrano podjetje, katerega temeljni proces je predmet preučevanja.
- Preučiti in predstaviti metodo za vrednotenje kakovosti razvoja programskih rešitev (ang. Capability Maturity Model Integration) in uporabiti njene faze zrelosti pri vrednotenju celotnega življenjskega cikla razvoja programske rešitve v izbranem podjetju.
- Predstaviti in preučiti življenjski cikel managementa poslovnih procesov in ga uporabiti v praktičnem delu kot osnovo za prenovo poslovnega procesa testiranja programskih rešitev.
- Predstaviti vlogo informacijske tehnologije pri digitalizaciji poslovanja prek treh temeljnih faz v povezavi z življenjskim ciklom managementa poslovnih procesov.
- Preučiti uporabo orodij MPP pri prenovi poslovnih procesov in uporabiti eno izmed njih – Bizagi Modeler in Bizagi Studio za prenovo temeljnega procesa v podjetju.
- Preučiti in predstaviti načine, kako podjetje izvede digitalizacijo poslovanja s pomočjo informacijske tehnologije.

Prvi del magistrskega dela je teoretični, drugi del pa je študija primera.

V prvem, teoretičnem delu sem uporabila metodološki pristop pregled tuje strokovne literature in pregled spletnih strani slovenskih ter tujih podjetij, ki so ponudniki orodij MPP in se ukvarjajo s prenovo ter digitalizacijo poslovanja.

V drugem, praktičnem delu pa sem kot raziskovalno metodo uporabila študijo primera. V podjetju sem popisala obstoječi temeljni poslovni proces – testiranje programskih rešitev – s pomočjo interne dokumentacije, obstoječih diagramov poteka procesa in s pomočjo pogovora z zaposlenimi, ki so vključeni v proces. Zastavila sem jim naslednji dve ključni vprašanji: katere aktivnosti opravljajo v procesu in koliko časa porabijo za izvajanje posameznih aktivnosti. Pri analiziranju obstoječega procesa sem preučila tudi uporabo aplikacije Jira, ki jo zaposleni uporabljajo za beleženje časa razvoja in testiranja programske rešitve. Metodologija pridobivanja podatkov je podrobneje opisana v tretjem poglavju magistrskega dela.

Magistrsko delo je sestavljeno iz štirih poglavij. Prvi dve poglavji sta teoretični, zadnji dve pa sta praktični, in sicer obravnavata prenovo in digitalizacijo temeljnega procesa v izbranem podjetju.

V prvem poglavju so opredeljeni in podrobneje opisani osnovni koncepti s področja poslovnih procesov, managementa poslovnih procesov, prenove poslovanja, Porterjeve verige vrednosti in Leavittovega diamanta. Opisana sta tudi življenjski cikel managementa poslovnih procesov v povezavi s prenovo poslovanja s pomočjo informacijske tehnologije prek treh temeljnih faz. V drugem delu poglavja so predstavljena orodja MPP – kakšne koristi in možnosti prinaša uporaba orodij MPP, na kakšen način naj podjetja izberejo primerno orodje MPP za njihovo prenovo poslovanja itd. Predstavljena je tudi metoda za vrednotenje kakovosti razvoja programskih rešitev.

V drugem poglavju je predstavljen koncept modeliranja poslovnih procesov, podrobneje je opisana Business Process Model and Notation (ang. Business process model notation, v nadaljevanju BPMN) in osnovne kategorije, ki jo sestavljajo. V nadaljevanju poglavja je primer poenostavljenega procesa, izrisanega v BPMN, pri katerem so uporabljeni osnovni gradniki za izris procesnega diagrama. Na koncu poglavja je predstavljen še programski paket Bizagi BPM Suite**.**

Tretje poglavje začenjam s predstavitvijo izbranega podjetja. Poglavje nadaljujem z opisom metodologije pridobivanja podatkov za potrebo prenove procesa testiranja programskih rešitev v izbranem podjetju. Pred opisom procesa testiranja, ki je tudi osrednja tema praktičnega dela magistrske naloge, predstavim in opišem življenjski cikel razvoja programskih rešitev, katerega opis je potreben za lažje razumevanje celotnega procesa testiranja programskih rešitev. Sledi predstavitev Porterjeve verige vrednosti v izbranem podjetju, kjer so predstavljeni vsi procesi, ki se odvijajo v izbranem podjetju. Pri vrednotenju kakovosti obstoječega procesa razvoja programskih rešitev uporabim metodo za vrednotenje kakovosti razvoja programskih rešitev, ki je podrobneje predstavljena v prvem poglavju magistrskega dela.

Pri prenovi in digitalizaciji procesa testiranja programskih rešitev uporabim življenjski cikel managementa poslovnih procesov, ki je teoretično opisan v prvem poglavju. Poglavje

nadaljujem z opisom in predstavitvijo obstoječega procesa testiranja programskih rešitev ter prenovljenega procesa testiranja programskih rešitev, in sicer v programu Bizagi Modeler v BPMN. Po predstavitvi modelov sledi tudi podrobnejša analiza obeh modelov in primerjava med njima. Nato sledi še diskusija o rezultatih analize in predlogi za izboljšavo obstoječega procesa.

V zadnjem, četrtem poglavju je predstavljena aplikacija, ki je rezultat samostojnega dela. Narejena je s pomočjo orodja Bizagi Studio, v katero je implementiran prenovljeni proces testiranja programskih rešitev v izbranem podjetju. Prikazanih je tudi nekaj primerov uporabe aplikacije. Na koncu poglavja sledi še diskusija o implementaciji prenovljenega procesa testiranja v izbrano podjetje.

Zadnjemu poglavju sledita še sklep in literatura.

1 OSNOVNI KONCEPTI S PODROČJA POSLOVNIH PROCESOV

V prvem poglavju so opredeljeni ključni pojmi s področja managementa poslovnih procesov. Najprej je podrobneje definiran poslovni proces, v nadaljevanju najdemo opis Porterjeve verige vrednosti in splošne opredelitve treh vrst procesov v podjetju. Nato sledi definiranje prenove poslovanja in navedba ključnih razlogov, zaradi katerih se podjetja odločijo za prenovo poslovnih procesov oziroma za celovito prenovo poslovanja. Poglavje se nadaljuje s podrobnejšo predstavitvijo managementa poslovnih procesov in življenjskega cikla managementa poslovnih procesov. Podpoglavje, v katerem je opisan življenjski cikel managementa poslovnih procesov, vključuje tudi opis prenove poslovanja s pomočjo informacijske tehnologije prek treh temeljnih faz. Poglavje nadaljujem s predstavitvijo modela za vrednotenje kakovosti procesa razvoja programskih rešitev, ki spada med procesne modele, katerega bom uporabila v praktičnem delu magistrskega dela. Ker je osrednji namen digitalizacija procesa testiranja programskih rešitev v izbranem podjetju, opišem in predstavim še orodja MPP. V zadnjem podpoglavju je opisan Leavittov diamant. Opredelitev Leavittovega diamanta je pomembna, saj se podjetja premalokrat zavedajo, da je v prenovo poslovanja potrebno vključiti vse socio–tehnične dejavnike. Ni namreč dovolj, da se prenova poslovanja izvede zgolj s pomočjo napredne informacijske tehnologije, ne da bi pri tem upoštevali tudi sociološki vidik prenove poslovanja.

Glavni namen prvega poglavja je predstavitev teorije, ki je uporabljena v praktičnem delu magistrskega dela.

1.1 Poslovni proces

V strokovni literaturi najdemo številne definicije, ki opredeljujejo poslovni proces. V nadaljevanju je navedenih nekaj definicij različnih avtorjev:

- Poslovni proces je skupek med seboj logično povezanih aktivnosti in nalog, ki se izvajajo, da podjetje doseže nek cilj. Poslovni procesi se izvajajo večkrat. Nekateri se izvajajo popolnoma ročno, nekateri se izvajajo deloma ročno, deloma avtomatizirano, nekateri pa se izvajajo popolnoma avtomatizirano (Weske, 2007, str. 4).
- »Proces ni prepoznaven le po aktivnostih, ki jih opravljajo njegovi izvajalci, pač pa predvsem po zaporedju dejavnosti in opravil, ki jih je potrebno izvesti, da bi na izhodni strani procesa dobili predvidene rezultate. Govorimo o ureditvi procesnih aktivnosti skozi čas in prostor, z začetkom in koncem ter jasno zaznanimi vhodi in izhodi.« (Kovačič, Jaklič, Indihar Štemberger & Groznik, 2004, str. 58–59).
- Devenport opisuje poslovni proces kot strukturiran niz dejavnosti, namenjenih ustvarjanju specifičnega proizvoda za določen trg ali stranko. Poslovni proces je sestavljen iz niza aktivnosti, ki omogočijo, da se vhodi pretvorijo v izhode (Vukšić–Bosilj, Hernaus & Kovačič, 2008, str. 16).
- Hammer in Champy sta postavila najpomembnejšo definicijo poslovnega procesa. Opisujeta ga kot skupek aktivnosti, ki pretvorijo enega ali več vhodov v izhod in s tem se ustvari proizvod ali storitev, ki predstavlja dodano vrednost za stranko (Vukšić–Bosilj, Hernaus & Kovačič, 2008, str. 16).

Če povzamem, je poslovni proces skupek dejavnosti in opravil, ki potekajo v določenem zaporedju tako, da se tekom procesa dani vhodi pretvorijo v izhod z namenom, da podjetje ustvari čim večjo dodano vrednost za stranko in si s tem zagotovi ustvarjanje dobička.

V podjetjih obstajajo tri vrste procesov (Jeston & Nelis, 2014, str. 94): temeljni procesi, podporni procesi in procesi upravljanja.

Temeljni procesi ustvarjajo največjo dodano vrednost za podjetje in za zunanje stranke podjetja. Ti procesi izhajajo iz primarnih dejavnosti v podjetju (Jeston & Nelis, 2014, str. 94–95). Temeljne procese razdelimo v tri skupine, ki so opisane v nadaljevanju:

- V prvo skupino se uvrščajo procesi snovanja, razvoja, ustvarjanja in predstavitve novega izdelka oziroma storitve (npr. v farmacevtski industriji je primer procesa snovanja in razvoja novih zdravil, katerega izhod je ideja za novo zdravilo).
- V drugo skupino se uvrščajo procesi proizvodnje izdelka ali pa ustvarjanje storitve (npr. v farmacevtski industriji je proces proizvodnje zdravil, katerega izhod je izdelano zdravilo).
- V tretjo skupino se uvrščajo procesi prodaje izdelka ali storitve (npr. v farmacevtskih podjetjih je to proces prodaje zdravil lekarnam, zdravstvenim ustanovam itd.).

Podporni procesi so namenjeni podpori temeljnim procesom v podjetju oziroma omogočajo izvajanje temeljnih procesov. Razlika med temeljnimi in podpornimi procesi je, da podporni procesi neposredno ne prinašajo dodane vrednosti za podjetje, ampak so pomembni pri samem poslovanju podjetja (Jeston & Nelis, 2014, str. 94). Ti procesi se odvijajo v oddelku informacijske tehnologije, v kadrovski službi, v računovodstvu, v

oddelku nabave, v skladišču itd. (Jeston & Nelis, 2014, str. 376).

Ključna razlika med temeljnimi in podpornimi procesi je, da so ključni odjemalci temeljnih poslovnih procesov zunanje stranke podjetja, ključni odjemalci podpornih procesov pa so zaposleni v podjetju oziroma notranje stranke podjetja.

Procesi upravljanja spremljajo in usmerjajo izvajanje temeljnih in podpornih procesov. Skrbijo, da se vsi procesi izvajajo v skladu z zakonodajo in predpisanimi pravili (Jeston & Nelis, 2014, str. 94). Spremljajo različne ključne kazalnike uspešnosti (ang. Key performance indicators) (tj. finančne, operativne) vseh procesov v podjetju in s tem omogočijo pregled nad tem, ali temeljni in podporni procesi sledijo strategiji podjetja oziroma izpolnjujejo zastavljene cilje poslovanja (npr. finančne). Primer procesa upravljanja je načrtovanje strategije poslovanja podjetja in spremljanje njenega izvajanja.

Pri poslovanju podjetja je pomembno, da so procesi, ki se izvajajo, učinkoviti in uspešni. Vse večje učinkovitosti in uspešnosti procesov podjetja ne bi mogla doseči brez uporabe napredne informacijske tehnologije.

O učinkovitosti procesa govorimo takrat, ko nas zanima čas izvedbe procesa in stroški, ki nastanejo z izvedbo procesa. Za merjenje učinkovitosti procesov se uporabljajo različni ključni kazalniki uspešnosti, s katerimi merimo porabo različnih virov, npr. količina porabljenega materiala, število zaposlenih, poraba finančnih virov itd. Učinkovitost procesov podjetja lahko dosežejo npr. z digitalizacijo poslovanja, odpravo nepotrebnih aktivnosti v procesih itd. (Kovačič, Jaklič, Indihar Štemberger & Groznik, 2004).

Drugi že omenjeni koncept, ki je prav tako pomemben pri poslovanju podjetja, je uspešnost procesa. Uspešnost procesa pomeni, da proces dela prave stvari. Lahko je nek proces učinkovit, ni pa uspešen, ali povedano drugače, podjetje lahko izvaja napačne stvari na učinkovit način. Uspešnost procesov podjetja dosežejo na različne načine, npr. s prenovo poslovanja, z novimi izdelki ali storitvami, prenovo poslovnega modela itd. Tako kot učinkovitost poslovnih procesov tudi uspešnost merimo s ključnimi kazalniki uspešnosti kot npr. zadovoljstvo strank, kakovost izdelkov in storitev itd. (Kovačič, Jaklič, Indihar Štemberger & Groznik, 2004).

Zgoraj je opisana splošna opredelitev treh vrst procesov v podjetju, v nadaljevanju pa si poglejmo, kako je Michael E. Porter opredelil poslovne procese v proizvodnem podjetju prek njegove verige vrednosti (Porter, 1998).

Podjetja si zastavijo strategijo in jo skušajo uresničevati z različnimi strateškimi načrti. Strategija je zastavljena tako, da njeno uresničevanje prinese podjetju čim večjo dodano vrednost in mu zagotavlja konkurenčno prednost na trgu. Ključno in najtežje vprašanje za vsako podjetje je, na kakšen način lahko doseže konkurenčno prednost. Leta 1985 je Michael Eugene Porter v svoji knjigi z naslovom Konkurenčna prednost opisal koncept verige vrednosti podjetja (Porter, 1998).

Porter je model verige vrednosti opredelil na podlagi analize vrednostne verige v proizvodnem podjetju, kot nabor dejavnosti, ki jih podjetje izvaja z namenom, da ustvari dodano vrednost za stranko in da mu omogoča ohranjati konkurenčno prednost na trgu. Porterjeva veriga vrednosti je zastavljena na način, da jo lahko uporabi vsako podjetje, ne glede na to, katera je njegova primarna dejavnost in v kateri panogi deluje (Porter, 1998, str. 36).

Porter (1998, str. 39) pravi, da se mora podjetje zavedati pomena dobro zastavljenih in optimalnih procesov, saj lahko le z njihovo pomočjo ustvari izdelek ali storitev, ki bo stranki predstavljala dodano vrednost in bo zanjo pripravljen plačati.

Porter je v proizvodnem podjetju razdelil procese v dve skupini, in sicer na temeljne in podporne. Med temeljne procese je uvrstil vhodno logistiko, proizvodnjo, izhodno logistiko, trženje, prodajo in podporo (poprodajne aktivnosti). Med podporne procese je uvrstil infrastrukturo, upravljanje s kadri, tehnološki razvoj in nabavo.

Temeljni procesi po Porterju so tisti, ki neposredno ustvarjajo dodano vrednost za stranko in podjetje ter podjetju omogočajo ohraniti konkurenčno prednost na trgu. Predstavljeni so v nadaljevanju (Porter, 1998, str. 39–40):

- Vhodna logistika: Zajema vse procese, ki vključujejo prevoz, skladiščenje in notranjo distribucijo surovin za proizvodnjo izdelka. Za učinkovite in uspešne procese vhodne logistike je pomembno, da podjetje v svoje procese vključi tudi dobavitelje in skupaj z njimi izboljša procese.
- Proizvodnja: Zajema vse procese, ki spreminjajo surovine v proizvode ali pa ustvarijo storitev. Povedano drugače, proizvodni procesi spreminjajo vhode v izhode.
- Izhodna logistika: Zajema vse procese, ki vključujejo skladiščenje, prevoz in distribucijo proizvedenih izdelkov.
- Trženje in prodaja: Zajema vse procese, ki vključujejo prodajo in promocijo storitev ter izdelkov, pa tudi vzdrževanje in navezovanje stikov s stranko.
- Podpora (poprodajne aktivnosti): Zajema vse procese, ki ohranjajo vrednost izdelka ali storitve in gradijo odnos med podjetjem ter stranko.

Podporni procesi po Porterju omogočajo, da se uspešno izvedejo temeljni procesi. Nekateri podporni procesi imajo zelo velik vpliv na učinkovitost in uspešnost poslovanja podjetja (Porter, 1998, str. 40–45).

- Infrastruktura podjetja: Zajema procese, ki se odvijajo v različnih oddelkih v podjetju in podpirajo delovanje celotne verige vrednosti v podjetju. Vrhnji management oblikuje strategijo delovanja podjetja skupaj z različnimi oddelki. Npr. sektor računovodstvo in finance načrtuje razporeditev finančnih virov, zmanjšanje stroškov itd.
- Upravljanje s kadri: Zajema procese, ki se odvijajo v kadrovski službi in so ključni pri usposabljanju ter izobraževanju že zaposlenih in novo zaposlenih. Procesi vključujejo

iskanje novih kadrov glede na potrebe podjetja, organizacijo izobraževanj in delavnic za zaposlene, vzpostavitev plačnega sistema, sistema nagrajevanja itd.

- Tehnološki razvoj: Zajema procese razvoja in vzdrževanja informacijske tehnologije. Ti procesi podjetju omogočajo, da razvija in vzdržuje svoje programske rešitve, ki nudijo učinkovito podporo poslovanju podjetja z najnaprednejšimi tehnološkimi rešitvami.
- Nabava: Zajema procese, ki se odvijajo v nabavi in so povezani z dobavitelji ter strankami. Primer procesov v nabavi so: nabava surovin za izdelke, pogajanje z dobavitelji glede cen surovin, iskanje novih dobaviteljev, obdelava naročil, načrtovanje zmanjševanja stroškov nabave itd.

Na sliki 1 je prikazana zgoraj opisana Porterjeva veriga vrednosti s temeljnimi in podpornimi aktivnostmi.

Slika 1: Porterjeva veriga vrednosti na primeru proizvodnega podjetja

Vir: Prirejeno po Porter, (1998, str. 37).

1.2 Management poslovnih procesov

Management poslovnih procesov (ang. Business process management, v nadaljevanju MPP) je področje, ki se je začelo razvijati v začetku devetdesetih let prejšnjega stoletja. Ukvarja se z modeliranjem, analizo, optimizacijo, avtomatizacijo, implementacijo, nadzorom in merjenjem učinkovitosti ter uspešnosti poslovnih procesov, s ciljem, da se podjetjem zagotovi uspešno poslovanje tako znotraj podjetja kot v njihovem poslovnem okolju. Prek MPP se zaposlene motivira, da se vključijo v proces izboljšave poslovanja (Jeston & Nelis, 2014, str. 9).

V strokovni literaturi najdemo številne definicije MPP. V nadaljevanju je navedenih nekaj definicij različnih svetovnih organizacij:

- Organizacija BPM Institute (2017) opredeljuje MPP kot učinkovito upravljanje s poslovnimi procesi od začetka do konca procesov. S pomočjo učinkovitega upravljanja s procesi ima podjetje jasnejšo strategijo in vizijo, optimalno razporejene in usklajene vire ter večjo disciplino pri vsakodnevnem izvajanju poslovnih procesov.
- Workflow Management Coalition (2018) opredeljuje MPP kot disciplino, ki vključuje katerokoli kombinacijo modeliranja, avtomatizacije, digitalizacije, izvedbe, merjenja in

optimizacije poslovnih procesov v podporo podjetjem, zaposlenim v podjetju, strankam ter partnerjem zunaj podjetja, da dosežejo zastavljene cilje.

- Gartner (2018), svetovno svetovalno in raziskovalno podjetje, ki svetuje podjetjem na različnih področjih poslovanja, opredeljuje MPP kot disciplino, ki uporablja različne metode za odkrivanje, modeliranje, analizo, merjenje, izboljšanje in optimizacijo poslovnih procesov. Poslovni proces usklajuje vedenje ljudi, sistemov in upravljanje z informacijami. Procesi so lahko strukturirani in ponovljivi ali nestrukturirani in spremenljivi. Za učinkovito in uspešno upravljanje s poslovnimi procesi se največkrat uporabljajo orodja MPP.

Uspešno delovanje MPP prinaša podjetjem naslednje prednosti (Dallas & Thandar Wynn, 2014): povečanje prihodkov in posledično dobička, zmanjšanje stroškov, optimizacijo delovne sile, izpopolnjevanje izdelkov in storitev, uporabo programskih rešitev, ki so prilagojene poslovanju podjetja, poveča se produktivnost zaposlenih itd. Prispeva pa tudi k boljšemu razumevanju potreb strank, zato se posledično izboljša odnos podjetje – stranka.

1.3 Prenova poslovnih procesov in prenova poslovanja

V današnjem poslovnem svetu preživijo le podjetja, ki so se sposobna hitro prilagajati konkurenčnemu okolju; ki razvijajo izdelke in storitve, ki strankam ponujajo dodano vrednost glede na konkurenčna podjetja; ki proizvajajo produkte po najnižji možni ceni v najkrajšem možnem času, obenem pa zagotavljajo največjo možno kakovost. Močna konkurenca in ogromna izbira izdelkov ter storitev strankam omogoča, da imajo pri izbiri ponudnika izdelka ali storitve večjo moč kot podjetja.

Gradišar, Jaklič & Turk (2007) navajajo, da v poslovnem svetu obstajajo trije prevladujoči dejavniki, ki podjetja prisilijo v prenovo poslovanja, če želijo na dolgi rok preživeti na trgu. Vsak izmed dejavnikov je opisan v nadaljevanju.

Vedno večje zahteve strank: Na trgu različna podjetja ponujajo podobne izdelke oziroma storitve. Ker so stranke vse bolj zahtevne, želijo diferencirane proizvode in storitve, ki jih lahko ponudijo le podjetja, ki imajo fleksibilno proizvodnjo ter nudijo visoko raven storitev.

Vedno večja konkurenca: Ni dovolj, da podjetja na trgu ponujajo izdelke in storitve, ki so cenovno sprejemljive za stranke. Cena je samo eden izmed dejavnikov nakupa. Stranke želijo imeti produkte, ki so visoko kakovostni, hitro dosegljivi in imajo primerno ceno. Na trgu preživijo podjetja, ki sledijo trendom najnovejše tehnologije, saj jim pri izdelavi izdelkov in ponudbi storitev tudi ta predstavlja konkurenčno prednost.

Nenehne spremembe: Zahteve strank so ob poplavi izdelkov in storitev vedno večje. Ker je njihova življenjska doba vse krajša, morajo podjetja nenehno razvijati nove izdelke in storitve ter pri tem uporabljati najnovejšo tehnologijo.

Različni avtorji različno opredelijo koncept prenove poslovnih procesov. V nadaljevanju je navedenih nekaj opredelitev različnih avtorjev:

- »Prenovo poslovnih procesov opredelimo kot temeljito preverjanje poslovnih procesov in njihovo korenito spremembo, ki jo sprožimo z namenom doseganja pozitivnih rezultatov na področjih, kot so zniževanje stroškov, povečanje kakovosti izdelkov ter storitev, skrajševanje dobavnih rokov in podobno.« (Kovačič, Jaklič, Indihar Štemberger & Groznik, 2004, str. 58).
- »Prenova poslovnih procesov je analiza in preoblikovanje delovnih procesov od začetka do konca, tako znotraj podjetja kot med podjetji, s ciljem, da se procese optimizira in avtomatizira na način, da bo njihovo izvajanje prineslo dodano vrednost podjetju.« (Chang, 2006, str. 23).
- Prenova poslovnih procesov temelji na razumevanju in korenitem preoblikovanju poslovnih procesov s pomočjo informacijske infrastrukture. Tako lahko podjetje s pomočjo prenovljenih procesov ustvari produkt, ki mu bo zagotovljena konkurenčna prednost na trgu ter bo lahko z njim ustvarilo čim več prihodkov in povečalo dobiček (Weske, 2007, str. 43).

V nadaljevanju je navedenih deset najpogostejših razlogov, zaradi katerih se podjetja odločijo za prenovo poslovanja. Razlogi so zbrani na podlagi izkušenj podjetnikov iz različnih panog, ki so se odločili za prenovo poslovanja (Peterson, Jaret & Findlay Schenck, 2016):

- Večanje stroškov, manjšanje prihodkov.
- Podjetje ne uresničuje zastavljene strategije in ne deluje v skladu z njo.
- Količina prodanih izdelkov ali storitev ne dosega zastavljenih prodajnih ciljev.
- Realno poslovanje podjetja se razlikuje od finančnih planov poslovanja.
- Zaposleni niso motivirani za delo, zmanjšuje se njihova produktivnost, vse več časa porabijo za rutinska dela in premalo časa za dela, ki podjetju dejansko prinesejo dodano vrednost.
- Projekti, ki so strateško pomembni za poslovanje podjetja, se ne izvedejo v časovnem roku, v času izvajanja projekta nastopijo različne težave, npr. pomanjkanje finančnih virov, pomanjkanje znanja zaposlenih itd.
- Na trgu se pojavi podjetje, ki predstavlja resno konkurenco obstoječemu podjetju.
- Glede na konkurenčna podjetja se v obstoječem podjetju uporablja stara tehnologija.
- Zaradi konkurenčnih podjetij stranke niso več lojalne blagovni znamki obstoječega podjetja.
- V podjetju zaradi hitre rasti poslovanja niso pripravljeni na nepričakovano povečanje povpraševanja, saj ne morejo proizvesti tolikšne količine izdelkov oziroma storitev kot je povpraševanja.

Temeljni cilji, ki jih želijo doseči podjetja s prenovo poslovnih procesov (Kovačič, Jaklič,

Indihar Štemberger & Groznik, 2004):

- Zniževanje stroškov in pri tem ohranjanje kakovosti izdelkov ter storitev.
- Dvig dodane vrednosti v vseh poslovnih procesih, ki se izvajajo v podjetju in izboljšanje kakovosti izdelkov ali storitev.
- Skrajšanje izvedbe vseh poslovnih procesov v podjetju in posledično znižanje stroškov poslovanja podjetja.
- Dvig zanesljivosti in doslednosti izvedbe poslovnih procesov ter posledično izboljšanje kakovosti izdelkov in izvedbe storitev.
- Osredotočanje na tiste dejavnosti v podjetju, ki predstavljajo konkurenčno prednost in prinašajo dodano vrednost. Dejavnosti, ki podjetju ne predstavljajo konkurenčne prednosti, je smiselno prepustiti v zunanje izvajanje podjetju, ki je specializirano za to dejavnost. (Npr. podjetje se odloči, da ne bo več imelo lastnega oddelka računovodstva, ker mu predstavlja prevelik strošek zaradi plač zaposlenih in plačevanja licence za računovodski program. Zato so se po analizi poslovanja in prenovi poslovnih procesov odločili, da bo računovodske storitve opravljal zunanji računovodski servis, saj je strošek zunanjega izvajalca manjši, kot če bi imelo podjetje lastno notranje računovodstvo.)
- Tesnejše sodelovanje z zunanjimi poslovnimi partnerji.

1.4 Življenjski cikel managementa poslovnih procesov

Preden se v podjetju lotijo prenove poslovanja, morajo imeti jasno razdelano in opredeljeno poslovno strategijo, poslovni model ter poslovne procese.

Poslovna strategija podjetja je nabor načrtov in usmeritev, s katerimi podjetje želi doseči zastavljene cilje. Z drugimi besedami lahko rečemo, da je strategija dolgoročno poslovno načrtovanje v obdobju treh do petih let (odvisno od podjetja, včasih je to obdobje še daljše). Ukvarja se s temeljnimi vprašanji poslovanja podjetja: ali je smiselno zgraditi nov proizvodni obrat; ali naj podjetje razvije nov izdelek ali storitev glede na trenutno stanje znotraj podjetja in zunanje poslovno okolje; ali naj se zaradi padca prihodkov odpusti večje število zaposlenih itd. Podjetje pri kreiranju strategije analizira konkurenčno okolje, notranje delovanje podjetja in prepoznava morebitne priložnosti ter grožnje v konkurenčnem okolju (Weske, 2017, str.17).

Poslovni model je model poslovanja organizacije oziroma načrt podjetja, kako ustvariti čim večje prihodke in zagotoviti čim manjše stroške. Poslovni model se oblikuje skladno s strategijo podjetja in vsebuje načrt kako bo podjetje doseglo uspešno poslovanje. Če gremo še globlje od poslovnega modela, pridemo do poslovnih procesov (Investopedia, 2018).

Poslovni proces sem že podrobno opredelila v prvem poglavju magistrskega dela, in sicer v razdelku 1.1.

Pred modeliranjem poslovnih procesov, ki predstavlja prvo fazo življenjskega cikla managementa poslovnih procesov, se mora znotraj podjetja oblikovati delovna skupina, katere člani sodelujejo pri prenovi poslovanja.

Vodja prenove je predstavnik ožjega vodstva podjetja. Njegova ključna naloga je, da skozi celotno prenovo poslovanja motivira vse udeležence in skuša odpraviti strah pred spremembami. Zagotavlja tudi pomoč managementu pri prenovi poslovanja (Gradišar, Jaklič & Turk, 2007).

Lastnik procesa je zadolžen za izvedbo in spremljanje poslovnega procesa, za opredelitev ciljev, ki jih želi doseči z njegovo izvedbo, za spremljanje rezultatov izvajanja poslovnega procesa ter odstopanj od zastavljenih ciljev izvedbe. Obstoječi proces usklajuje z drugimi procesi, odgovoren je za imenovanje vodij znotraj procesa, zadolžen je za načrtovanje proračuna izvedbe celotnega procesa, prizadeva si za nenehno izboljšanje poslovnega procesa, spremlja usposobljenost zaposlenih in kakovost izvajanja procesa (Kovačič, Jaklič, Indihar Štemberger & Groznik, 2004).

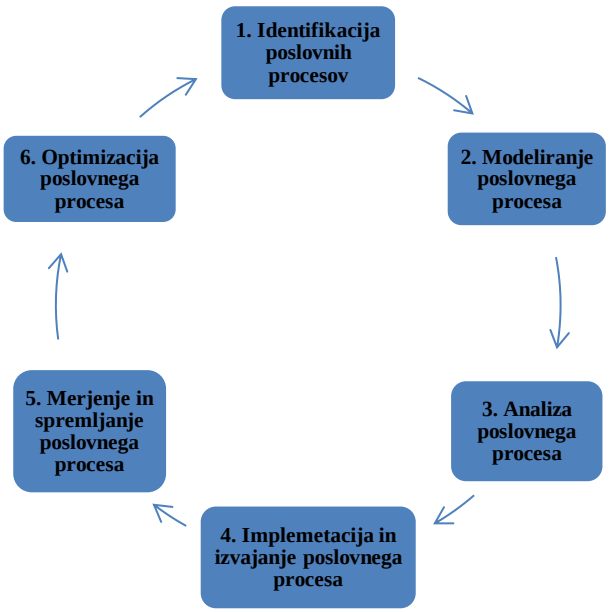
Poslovni analitik je običajno zaposlen v oddelku informacijske tehnologije. Pri prenovi poslovanja tesno sodeluje z lastniki procesa in s programerji, ki razvijajo informacijske rešitve za digitalizacijo poslovnih procesov ter nudijo podporo pri njihovi uporabi. Ključna naloga poslovnega analitika je, da s pomočjo dokumentacije, obstoječe aplikacije (v katero je integriran obstoječi proces), lastnika procesa in ostalih ključnih udeležencev analizira obstoječe poslovne procese, skuša opaziti njihove pomanjkljivosti in predlaga izboljšave. Poslovni analitik je vključen v celotni življenjski cikel managementa poslovnih procesov in nenehno išče izboljšave v procesih, ki se odvijajo v podjetju (Pratt & White, 2018).

Predstavniki iz oddelka informacijske tehnologije so zadolženi, da razvijejo takšne informacijske rešitve, ki bodo podprle in digitalizirale poslovne procese. Pri oblikovanju informacijskih rešitev sodelujejo z lastniki procesov in poslovnimi analitiki. Nenehno sledijo trendom naprednih informacijskih rešitev, ki bi jih želeli vključiti v podjetje (Jeston & Nelis, 2014, str. 22).

Življenjski cikel managementa poslovnih procesov sestavlja šest zaporednih faz, ki so prikazane na sliki 2 (Križevnik & Branko Jurič, 2009):

- 1. faza: identifikacija poslovnih procesov,
- 2. faza: modeliranje poslovnega procesa,
- 3. faza: analiza poslovnega procesa,
- 4. faza: implementacija in izvajanje poslovnega procesa,
- 5. faza: merjenje in spremljanje poslovnega procesa in
- 6. faza: optimizacija poslovnega procesa.

Slika 2: Življenjski cikel managementa poslovnih procesov



Vir: Winkler (2018).

– 1. faza: identifikacija poslovnih procesov

Identifikacija poslovnih procesov je prva faza življenjskega cikla managementa poslovnih procesov. Cilj prve faze je identificirati in opredeliti kateri procesi sploh obstajajo v podjetju ter kako so med seboj povezani. Rezultat identifikacije poslovnih procesov je nova ali posodobljena procesna arhitektura, ki omogoča pregled vseh procesov v podjetju in pojasnjuje, kako so med seboj povezani (Dumas, La Rosa, Mendling & Reijers, 2013, str. 22–23).

– 2. faza: modeliranje poslovnega procesa

Modeliranje poslovnih procesov je druga faza življenjskega cikla managementa poslovnih procesov. Cilj druge faze je izdelati modele obstoječih poslovnih procesov (ang. AS–IS model, v nadaljevanju AS–IS model). Pred fazo modeliranja mora delovna skupina s pomočjo intervjujev pridobiti čim več informacij o obstoječih procesih (npr. katere aktivnosti se izvajajo v posameznem poslovnem procesu, kakšni so rezultati posameznega procesa, koliko zaposlenih je vključenih v proces in katere aktivnosti izvaja določeni zaposleni v poslovnem procesu, kateri dokumenti so vključeni v določen proces, kakšno je zaporedje izvajanja aktivnosti itd.) (Križevnik & Branko Jurič, 2009).

Zelo pomembno je, da se v fazi modeliranja izriše AS–IS model obstoječega procesa, ki je odraz dejanskega stanja in ne vključuje morebitnih idej za izboljšavo poslovnega procesa. Procesi, ki so kompleksnejši, jih je zaradi lažjega razumevanja bolje izrisati s podprocesi. V drugi fazi se odkrivajo ozka grla v trenutno izvajanih procesih, natančno se opredeli obremenjenost virov, analizira se uporaba trenutnih informacijskih rešitev itd. (Weske,

2007, str. 12).

– 3. faza: analiza poslovnega procesa

Analiza je tretja faza življenjskega cikla managementa poslovnih procesov. Cilj analize je izvesti validacijo, simulacijo, časovno analizo in analizo virov AS–IS modela. Glede na rezultate analize AS–IS modela prepoznajo v podjetju možnosti za izboljšavo obstoječega procesa. Priporočljivo je, da se simulacija procesnega modela naredi za različno število iteracij procesa in za različno število razpoložljivih virov (Križevnik & Branko Jurič, 2009).

– 4. faza: implementacija in izvajanje poslovnega procesa

Implementacija in izvajanje je četrta faza življenjskega cikla managementa poslovnih procesov. V zadnjih letih si podjetja prizadevajo, da bi bili poslovni procesi in informacijske rešitve kar najbolj medsebojno povezani. To pomeni, da v podjetju za poslovne procese razvijejo prilagojeno informacijsko rešitev ali pa da na trgu kupijo že obstoječo programsko rešitev, ki ustreza poslovnim procesom (Križevnik & Branko Jurič, 2009). Lastni razvoj aplikacij, ki bi bile prilagojene procesom, je drag in dolgotrajen, saj lahko traja tudi več let. Zato podjetja za podporne procese, ki se izvajajo npr. v kadrovski službi, nabavi, skladišču itd. kupijo na trgu že obstoječe programske rešitve, npr. ERP, ki so v povprečju cenejše od razvoja lastnih programskih rešitev. ERP so uporabni predvsem za podporne procese, saj se ti procesi bistveno ne razlikujejo med podjetji. ERP rešitve pokrijejo slabih štirideset odstotkov vseh procesov, ki se izvajajo v podjetjih (Moore in drugi, 2017, str. 75).

– 5. faza: merjenje in spremljanje poslovnega procesa

Merjenje in spremljanje poslovnih procesov je peta faza življenjskega cikla managementa poslovnih procesov. V tej fazi v podjetju definirajo ključne kazalnike uspešnosti za merjenje in spremljanje poslovnih procesov, to so npr. čas obrata zalog, stroški zalog, čas izvedbe naročila, število pravočasno dostavljenih pošiljk, zadovoljstvo strank, kakovost izdelkov in storitev itd. (Križevnik & Branko Jurič, 2009). Smiselno je, da je čim več ključnih kazalnikov uspešnosti takšnih, ki omogočajo spremljanje in merjenje poslovnega procesa v realnem času. S podatki, ki jih v podjetju pridobijo prek ključnih kazalnikov uspešnosti, v podjetju izdelujejo statistična poročila, z njihovo pomočjo pa ocenijo uspešnost in učinkovitost izvajanih procesov (Xiao, 2016).

– 6. faza: optimizacija poslovnega procesa

Optimizacija poslovnih procesov je šesta in hkrati zadnja faza življenjskega cikla managementa poslovnih procesov. Cilj zadnje faze je izdelati model prenovljenega poslovnega procesa (ang. TO–BE model, v nadaljevanju TO–BE model) poslovnega procesa. V veliko pomoč pri izdelavi modela so ključni kazalniki uspešnosti, ki so bili

opredeljeni v peti fazi (Križevnik & Branko Jurič, 2009).

Optimizacija prinese nekaj ključnih prednosti pri poslovanju podjetja (Križevnik & Branko Jurič, 2009):

- Podjetje ima po optimizaciji boljšo preglednost nad izvajanjem procesov. Lahko se zgodi, da določeni deli procesov, ki so bili del starih procesov, postanejo po optimizaciji samostojni. Določeni procesi se poenostavijo, pri drugih pa se izrazi večja kompleksnost.
- Poveča se učinkovitost izvajanja notranjih (znotraj podjetja) in zunanjih (javnih, ki se odvijajo na relaciji podjetje – poslovni partnerji) poslovnih procesov.
- Zaradi boljšega pregleda nad procesom se izboljša obvladovanje napak.
- Zaradi boljše izkoriščenosti virov se znižajo stroški poslovanja, poveča se prodaja izdelkov in storitev.
- Zaradi izboljšanega poslovanja se posledično poveča zadovoljstvo strank podjetja (npr. zaradi prenovljenih procesov v oskrbni verigi podjetje skrajša dobavni čas pošiljke, zmanjša se delež uničenih produktov itd.).

Po fazi optimizacije je smiselno, da v podjetju ustvarijo ključne kazalnike uspešnosti, s katerimi merijo uspešnost in učinkovitost optimiziranih procesov. Priporočljivo je tudi, da pri optimiziranih procesih izvedejo simulacije in tako dejansko opazijo spremembo med starim in novim – optimiziranim procesom (Jeston & Nelis, 2014, str. 42–43).

Težave, ki nastopijo v fazi optimizacije procesa so naslednje (Križevnik & Branko Jurič, 2009):

- Zadolženi za izboljšavo procesa se velikokrat osredotočijo zgolj na odpravo ozkih grl v procesu in premalokrat predlagajo konkretne izboljšave.
- Pri uvedbi novih procesov se zaposlenim predstavi prednosti izboljšanih procesov in se jim ponudi pomoč pri uporabi novih programskih rešitev. Le na ta način bodo zaposleni imeli bolj pozitiven odnos do sprememb, ki so prišle s prenovo poslovnih procesov. Zmotno je prepričanje, ki ga zasledimo v današnjem poslovnem svetu, da je za podjetje pomembno le, da uporablja sodobne informacijske rešitve, ki mu omogočajo digitalizacijo poslovnih procesov, saj zgolj to ni dovolj.
- Podjetja pogosto želijo dobre prakse poslovanja drugih podjetij prenesti v svoje poslovanje. Vsako podjetje se mora zavedati, da če je nekaj dobro za eno podjetje, ni nujno, da bo dobro tudi za drugo.
- Velika težava nastopi, če se pri optimiziranih procesih opredeli napačne ključne kazalnike uspešnosti. V tem primeru v podjetju ne morejo realno oceniti uspešnosti in učinkovitosti novega procesa.

Podjetja se dandanes lotijo prenove poslovanja večinoma s pomočjo informacijske tehnologije. Attaran (2003) opisuje, kako zelo pomembno je, da podjetja v celotni

življenjski cikel managementa poslovnih procesov pravilno vključijo vlogo informacijske tehnologije prek treh temeljnih faz:

- 1. faza: procesi še niso zasnovani.
- 2. faza: izdelovanje načrta in izvedbe prenove ter digitalizacije poslovanja tako s tehnološkega kot socialnega vidika.
- 3. faza: zaključek razvoja informacijskih rešitev in spremljanje rezultatov.

Vsaka faza je podrobneje opisana v nadaljevanju.

Če podjetje želi izvesti celovito prenovo poslovanja ali pa prenovo določenih procesov s pomočjo uporabe življenjskega cikla managementa poslovnih procesov in informacijske tehnologije, je pomembno, da faze prenove s pomočjo informacijske tehnologije v podjetju pravilno vključijo v življenjski cikel managementa poslovnih procesov.

Pri opisu vsake faze prenove poslovanja s pomočjo informacijske tehnologije je navedena faza življenjskega cikla managementa poslovnih procesov, v katero jo je smiselno vključiti:

- 1. faza: procesi še niso zasnovani

V prvi fazi je pomembno zavedanje podjetja, da ne bo doseglo zastavljene stopnje digitalizacije poslovanja, če ima pomanjkljivo oziroma slabo zastavljeno strategijo in vizijo. Prek dobro opredeljene strategije in vizije mora namreč podjetje natančno opredeliti in razdelati poslovne procese. Primer dobre prakse je podjetje Walmart, kjer so pred prenovo procesov v oskrbni verigi jasno opredelili strategijo podjetja in analizirali distribucijo blaga v prodajalne. Ugotovili so, da lahko zmanjšajo stroške in zagotovijo dodano vrednost stranki, če v svoje prodajalne vključijo tudi dobavitelje. Informacijske rešitve so Walmartu omogočile digitalizirati oskrbno verigo. Informacijski sistem so razvili tako, da so bile med seboj povezane vse njihove trgovine in vsa skladišča (Attaran, 2003).

Zaradi vse večje globalizacije se čedalje več podjetij odloča, da bo širilo svoje poslovanje na tuje trge (bodisi v tuje države ali pa celo na druge kontinente). V tujini tako odpirajo poslovne enote. Pri vodenju in nadziranju se zaradi geografske razpršenosti posameznih enot lahko pojavijo težave pri vzpostavljanju komunikacije med poslovnimi enotami in sedežem podjetja, Posledično lahko pride do asimetričnih oz. nepopolnih informacij o poslovanju poslovnih enot. S pomočjo sodobne informacijske tehnologije so premagane geografske ovire, izboljša se izmenjava informacij med poslovnimi enotami in sedežem podjetja. Primer dobre prakse je podjetje General Electric, ki jim informacijski sistem omogoča virtualna srečanja poslovnih partnerjev iz tujine, delitev načrtov poslovanja in izvedbo obsežnejših analiz poslovanja celotne organizacije (Attaran, 2003).

Prenova in digitalizacija poslovanja podjetja morata težiti k procesno usmerjeni organizaciji. Le s sodelovanjem med zaposlenimi v različnih oddelkih v podjetju se lahko

doseže uspešno prenovo in digitalizacijo poslovanja. Izmenjava informacij med zaposlenimi je ključnega pomena za nadgradnjo, prenovo, oceno uporabnosti trenutnega informacijskega sistema in idejno zasnovo, kako naj bi izgledal bodoči informacijski sistem (Barjis, 2008).

Digitalizacija poslovanja prinaša tudi nova delovna mesta. Ne samo v podjetjih, kjer razvijajo napredne informacijske rešitve, ampak tudi v podjetjih, v katerih oddelek informacijske tehnologije predstavlja podporno funkcijo v podjetju (Attaran, 2003). Uporaba napredne in hitro spreminjajoče informacijske tehnologije sili zaposlene tudi izven oddelka informacijske tehnologije, da pridobijo čedalje več tehničnega znanja. Npr. zaposleni v marketingu in prodaji morajo imeti poleg znanja s svojega strokovnega področja tudi znanja, ki jim omogočajo pridobiti in obdelati podatke iz različnih aplikacij (Araujo, 2017).

Prvo fazo prenove poslovanja s pomočjo informacijske tehnologije, ko procesi še niso zasnovani, naj sovpada z opredelitvijo strategije in vizije podjetja in poslovnega modela ter prvo fazo življenjskega cikla managementa poslovnih procesov (tj. identifikacija poslovnih procesov). V podjetju pa naj opredelijo tudi kaj želijo doseči s pomočjo digitalizacije poslovnih procesov.

- 2. faza: izdelovanje načrta in izvedbe prenove ter digitalizacije poslovanja tako s tehnološkega kot socialnega vidika

V drugi fazi v podjetju preučujejo vplive prenove in digitalizacije poslovanja z družbenega vidika ter izvedejo celotno prenovo in digitalizacijo poslovanja. Pred samim načrtovanjem informacijskih rešitev se je potrebno posvetiti zaposlenim. Opredeli se procese in aktivnosti v procesih, ki jih izvajajo zaposleni. Vrhnji management mora skupaj s kadrovskim oddelkom načrtovati preoblikovanje delovnih mest in nalog ter ustvariti nova delovna mesta, ki so nujno potrebna za obstoj in konkurenčno prednost podjetja. Zaposlene mora seznaniti s prednostmi in morebitnimi težavami prenove ter digitalizacije poslovanja, jih aktivno vključiti v proces prenove poslovanja in skupaj z njimi oblikovati predloge za digitalizacijo poslovanja (Attaran, 2003).

Kot že omenjeno, lahko digitalizacijo poslovanja podjetje izvede interno, s pomočjo razvoja lastnih programskih rešitev ali pa za to najame zunanje podjetje, ki nudi svetovanje in razvoj informacijskih rešitev.

Koristi uporabe informacijske tehnologije pri prenovi poslovanja so naslednje (Grover, Jeong Ryul, Kettinger & T. C. Teng, 2015):

- Z uporabo naprednih orodij, ki omogočajo spremljanje procesa razvoja informacijskih rešitev, in s projektnim vodenjem se izboljša komunikacija med vsemi udeleženci prenove poslovanja – od zaposlenih, ki so vključeni v proces in bodo uporabljali nove informacijske rešitve do programerjev, ki razvijajo informacijske rešitve.

- Celotni življenjski cikel poslovnega procesa lahko zajamemo z informacijskimi rešitvami, kar omogoča zaposlenim v organizaciji nenehno spremljati poslovne procese tudi po digitalizaciji poslovanja.
- Uporaba naprednih informacijskih rešitev omogoča boljši pregled izvajanja poslovnih procesov (npr. enotna baza podatkov omogoča zaposlenim iz različnih oddelkov dostop, dopolnjevanje in spreminjanje podatkov na enem mestu. Pretok informacij je hitrejši, podatki so pravilnejši, vsem zaposlenim je dostop do baze podatkov mogoč v vsakem trenutku itd.).
- Rešitve v oblaku omogočajo hrambo večjih količin podatkov, hrambo papirnatih dokumentov v elektronski obliki, možnost hrambe pomembnih dokumentov v elektronski obliki pri zunanjem ponudniku tovrstnih storitev itd.
- V podjetju se poveča učinkovitost in uspešnost poslovanja.

Druga faza (izdelovanje načrta in izvedbe prenove in digitalizacije poslovanja tako s tehnološkega kot socialnega vidika) naj sovpada z drugo (tj. modeliranje poslovnega procesa), tretjo (tj. analiza poslovnega procesa), četrto (tj. implementacija in izvajanje poslovnega procesa), peto (tj. merjenje in spremljanje poslovnega procesa) in šesto fazo (tj. optimizacija poslovnega procesa) življenjskega cikla managementa poslovnih procesov.

- 3. faza: zaključek razvoja informacijskih rešitev in spremljanje rezultatov

Tretja faza je najzahtevnejši del, saj je potrebno zaposlene usposabljati za uporabo novih informacijskih rešitev, spremeniti njihov odnos – predvsem starejših zaposlenih – do uvedbe sprememb v poslovanju. Z novimi rešitvami se morajo soočiti zaposleni, pa tudi vodstvo, ki naj bi po uvedbi novih informacijskih rešitev zaposlenim nudilo tehnično in vsebinsko podporo (Attaran, 2003).

Na novo je potrebno definirati kazalce uspešnosti in učinkovitosti poslovanja, potrebno je spremljati napredek zaposlenih po izvedbi digitalizacije in slediti strategiji ter viziji podjetja.

Vloga napredne informacijske tehnologije v podjetju (Attaran, 2003):

- Nekatere aktivnosti v procesih se izvajajo popolnoma avtomatizirano.
- Lažje nadzorovanje poslovanja in hitrejše odkrivanje ozkih grl.
- Vpogled v trenutno poslovanje v vsakem trenutku.
- Sodobna informacijska tehnologija omogoča podjetjem, ki imajo geografsko razpršene poslovne enote, lažjo komunikacijo med zaposlenimi iz različnih držav oz. kontinentov, tako da pretok informacij med zaposlenimi poteka v realnem času prek različnih elektronskih naprav (mobilne naprave, tablični računalniki, prenosni računalniki itd.).
- Zaradi razvoja naprednih informacijskih rešitev se nekateri pomembni dogodki v podjetjih odvijajo kar prek spleta, npr. izvajanje video konferenc, kjer lahko poteka glasovanje o pomembnih odločitvah za podjetje, vodenje projektov, spremljanje

napredka projektov, delitev pomembnih dokumentov med zaposlenimi, izvajanje srečanj s poslovnimi partnerji iz različnih držav itd.

Zadnja, tretja faza (zaključek razvoja informacijskih rešitev in spremljanje rezultatov), naj sovpada s šesto fazo življenjskega cikla managementa poslovnih procesov (tj. optimizacijo poslovnega procesa).

Zadnja faza v razvoju informacijskih rešitev predstavlja eno težjih preizkušenj za vodstvo podjetja, katerega naloga je, da pomaga zaposlenim čim hitreje osvojiti uporabo novih informacijskih rešitev in sprejeti spremembe v procesih kot pozitivne.

1.5 Zrelostni model za vrednotenje kakovosti razvoja programske opreme

Pri razvoju programskih rešitev vplivajo na kakovost končnega izdelka trije ključni dejavniki: zaposleni, ki so vključeni v proces razvoja programskih rešitev, tehnologija, s pomočjo katere razvijajo programsko rešitev in zastavljeni proces razvoja programske rešitve (Majumdar, Ashique–Ur–Rouf, Islam & Arefeen, 2011).

Programske rešitve postajajo z leti čedalje bolj kompleksne, zato mora biti celotni proces razvoja programske rešitve dobro zastavljen. Naročniki programskih rešitev pričakujejo od podjetij, ki razvijajo programske rešitve, produkte in storitve v najkrajšem možnem času, po konkurenčni ceni ter da funkcionalnosti v programski rešitvi delujejo brezhibno. Podjetja, ki razvijajo programske rešitve, pri razvoju programskih rešitev naletijo na različne težave. Npr. pri testiranju programske rešitve ugotovijo, da je število napak v funkcionalnostih večje od pričakovanih, kar posledično podaljša časovni rok predaje programske rešitve naročniku (Majumdar, Ashique–Ur–Rouf, Islam & Arefeen, 2011).

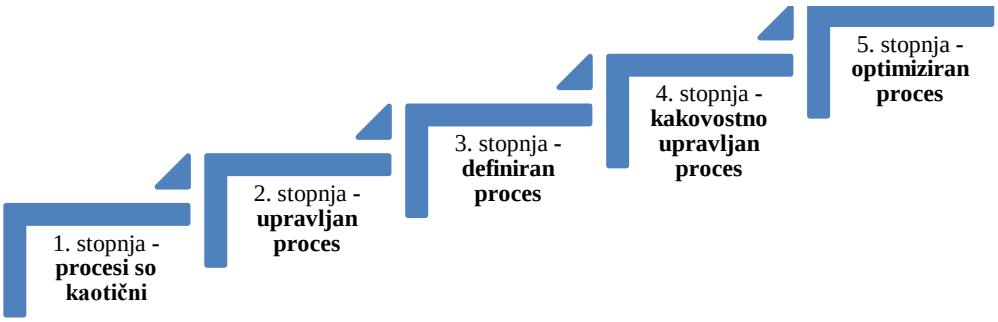
Velik vpliv na kakovost same programske rešitve ima celotni razvojni proces programske rešitve. Na trgu obstajajo številni procesni modeli, ki omogočajo podjetjem izboljšati izvajanje obstoječih procesov. Eden takšnih modelov je zrelostni model za vrednotenje kakovosti razvoja programske opreme (ang. Capability Maturity Model Integration, v nadaljevanju CMMI) (Lewis, 2005, str. 24–25).

CMMI je procesni model, ki opisuje najboljše primere praks za upravljanje, merjenje in spremljanje procesov razvoja programskih rešitev. Opisuje značilnosti dobrih procesov in narekuje smernice podjetjem, ki sama načrtujejo in oblikujejo procese. Model temelji na ideji, da so procesi – prav tako kot tehnologija – pomembni za kakovostni razvoj programske rešitve (Rožanc & Mahnič, 2003).

Model CMMI je predstavljen kot petstopenjski model zrelosti, prikazan je na sliki 3. Vsaka stopnja v modelu natančno definira značilnosti procesa, ki jih mora podjetje izpolnjevati, če želi doseči določeno stopnjo zrelosti. Za vsako zrelostno stopnjo so določena naslednja procesna področja: splošni cilji (ang. generic goals), specifični cilji (ang. specific goals),

splošne prakse (ang. generic practices) in specifične prakse (ang. specific practices) (Rožanc & Mahnič, 2003).

Slika 3: Zrelostni model za vrednotenje kakovosti razvoja programske opreme



Vir: Povzeto in prirejeno po ITFAAT, (2018).

– 1. stopnja zrelosti: procesi so kaotični

Na tej stopnji zrelosti so procesi kaotični in se oblikujejo glede na dane projekte, naloge itd. Uspeh podjetij, ki imajo procese v tej fazi, je odvisen zgolj od sposobnosti zaposlenih in moči posameznikov in ne od dobro organiziranih in razdelanih procesov (Rožanc & Mahnič, 2003). Projekti v takšnih podjetjih pogosto presežejo zastavljen časovni okvir in proračun. Ko nastopi čas krize, so v podjetju pripravljeni opustiti dejavnost, saj ne bi mogli ponovno doseči preteklih poslovnih uspehov, ki so jih dosegli s pomočjo nekaterih projektov (Beth Chrissis, Konrad & Shrum, 2011, str. 42).

– 2. stopnja zrelosti: upravljan proces

Na tej stopnji imajo v podjetju vzpostavljene osnovne procese vodenja projektov. Procese načrtujejo in jih spremljajo, ravno tako merijo in spremljajo stroške, časovni okvir razvoja programske rešitve in učinkovitost dela (Majumdar, Ashique–Ur–Rouf, Islam & Arefeen, 2011). Procesi so vzpostavljeni tako, da jih je mogoče ponoviti na podobnih projektih, kjer želijo doseči podobne rezultate. Končni izdelek ali storitev zadosti zahtevam, ciljem in standardom podjetja (Beth Chrissis, Konrad & Shrum, 2011, str. 43).

– 3. stopnja zrelosti: definiran proces

Od druge stopnje zrelosti se tretja razlikuje v tem, da so procesi strožje in bolj natančno definirani, med seboj se povezujejo in dopolnjujejo. Natančno so definirani vhodi in izhodi, aktivnosti, naloge in vloge zaposlenih. Procesi so standardizirani, popisani in dokumentirani. Proces projekta se izvede s pomočjo prilagoditve standardnih procesov, ki so v podjetju popisani. Učinkovitost in kakovost procesov se meri kvalitativno (Beth Chrissis, Konrad & Shrum, 2011, str. 43).

– 4. stopnja zrelosti: kakovostno upravljan proces

Na tej stopnji v podjetju poleg kvalitativnih kazalnikov opredelijo še kvantitativne. Z njimi se spremlja in meri kakovost ter učinkovitost procesov razvoja izdelkov in storitev. Za merjenje procesov se zbirajo večje količine kvantitativnih podatkov, ki se s pomočjo statističnih orodij tudi obdelajo (Majumdar, Ashique–Ur–Rouf, Islam & Arefeen, 2011). Vsaka sprememba, ki se zgodi v procesu, se natančno definira in kvantitativno izmeri. Za spremembe v procesih se natančno opredelijo vzroki. Od tretje stopnje zrelosti se četrta razlikuje v tem, da se kakovost in uspešnost izvedenih procesov meri tudi kvantitativno (ne le kvalitativno kot v tretji stopnji) in da je izvedba procesov na tej stopnji bolj predvidljiva. Opredelijo se podrobnejši cilji, kot so zadovoljstvo naročnikov in končnih uporabnikov programskih rešitev (Beth Chrissis, Konrad & Shrum, 2011, str. 43).

– 5. stopnja zrelosti: optimiziran proces

Ko podjetje doseže to stopnjo zrelosti, teži k nenehnemu izboljševanju procesov in se je sposobno uspešno ter učinkovito prilagoditi spremembam poslovnega okolja z inovativnimi spremembami poslovnih procesov. Po uvedbi sprememb na področju tehnologije in aktivnosti v procesih podjetje nenehno spremlja ter meri spremembe, ki vplivajo na celotni proces. Na tej stopnji je podjetje sposobno oceniti in ovrednotiti vpliv sprememb na procese. Optimizacija procesov, ki je prilagojena strategiji, poslovnim vrednotam podjetja in ciljem organizacije, se izvede v sodelovanju z zaposlenimi in z uporabo napredne informacijske tehnologije (Beth Chrissis, Konrad & Shrum, 2011, str. 44).

V praktičnem delu magistrskega dela bom uporabila model CMMI za ocenitev trenutnega stanja procesa razvoja programskih rešitev, ki ga trenutno uporabljajo v izbranem podjetju.

1.6 Orodja za podporo managementu poslovnih procesov

Orodja MPP v podjetjih uporabljajo za modeliranje, analizo, merjenje ključnih kazalnikov uspešnosti, optimizacijo, avtomatizacijo in digitalizacijo poslovnih procesov (Finances Online reviews for business, 2018a). So zelo pomembna pri izboljšanju poslovanja, nudijo podporo lastnikom procesov, saj omogočajo boljši pregled nad celotno izvedbo procesa. Organizatorji procesa lahko optimalno razporedijo delovne naloge zaposlenim in načrtujejo finančna sredstva, ki so potrebna za izvedbo procesa. S pomočjo različnih ključnih kazalnikov uspešnosti lahko spremljajo rezultate izvedenega procesa (npr. čas izvedbe ene iteracije ali več iteracij poslovnega procesa, število porabljenih finančnih virov, število zaposlenih, ki so udeleženi v procesu itd.). Glede na to, kakšne ima podjetje potrebe po upravljanju s poslovnimi procesi, obstaja na trgu veliko različnih orodij MPP, ki vsebujejo različne platforme – od osnovnih do inteligentnih platform (Appian, 2018).

V nadaljevanju so opisane tri vrste platform orodij MPP, ki jih je opredelila svetovna

svetovalna organizacija Gartner (2018):

Osnovne MPP platforme (ang. Basic BPM Platforms). Osnovne platforme vsebujejo tista orodja MPP, ki so namenjena modeliranju, validaciji in analizi poslovnih procesov. Ta orodja imajo enostavnejši, uporabniku prijazen grafični vmesnik in ga navadno uporabljajo zaposleni, ki nimajo naprednejšega tehničnega znanja. Takšna orodja uporabljajo manjša in srednja podjetja, ki nimajo kompleksnejših poslovnih procesov (Software Advice, 2018).

Naprednejše MPP platforme (ang. BPM Suites – BPMs). Naprednejše platforme vsebujejo naprednejša orodja MPP in večje število funkcionalnosti ter od uporabnika zahtevajo naprednejše poznavanje orodij MPP. Uporabljajo jih zaposleni v oddelku informacijske tehnologije in zaposleni v drugih oddelkih podjetja, ki imajo naprednejše znanje s področja informacijske tehnologije ter poznavanja orodij MPP. Orodja MPP z naprednejšimi platformami so primerna za podjetja, ki želijo integracijo poslovnih procesov v aplikacije, kot npr. ERP sisteme, CRM–je itd. S tem želijo doseči večje dopolnjevanje in usklajevanje med procesi, ki se odvijajo v različnih oddelkih podjetja. Primerna so predvsem za večja podjetja, ki imajo kompleksnejše in obsežnejše poslovne procese (Software Advice, 2018).

Inteligentne MPP platforme (ang. Intelligent BPM Suites – iBPMS). Inteligentne MPP platforme vsebujejo najnaprednejša orodja MPP, ki vsebujejo vse tiste funkcionalnosti, ki jih imajo orodja MPP z osnovnimi in naprednimi platformami. Poleg tega pa omogočajo tudi razvoj samostojnih aplikacij, v katere je možno integrirati poslovne procese, in s tem doseči digitalizacijo poslovanja podjetja. Glede na zahtevnost uporabe orodij MPP, ki vsebujejo inteligentne platforme, jih uporabljajo zaposleni iz oddelka informacijske tehnologije za razvoj aplikacij (Software Advice, 2018). Aplikacije omogočajo vpogled v potek poslovnega procesa v realnem času in so zmožne same izboljšati ter nadgraditi procese. Primer takšnega orodja je orodje Bizagi (Gartner, 2018).

Ključne prednosti, ki jih prinašajo orodja MPP, so naslednje (Finances Online reviews for business, 2018b):

- Zmanjšanje stroškov.
- Povečanje učinkovitosti in uspešnosti izvedbe procesov.
- Zagotavljanje večje skladnosti izvedbe procesov z zakoni, predpisi in poslovnimi pravili.
- Z njihovo uporabo se poveča poslovna agilnost.
- Poveča se konkurenčna prednost podjetja na področju načina poslovanja.
- Spodbujanje sodelovanja s strankami, saj na ta način podjetje pridobi podatke o preferencah, potrebah in željah svojih strank.
- Povečanje produktivnosti zaposlenih.
- Omogočajo optimalno razporeditev virov.
- Omogočajo izvedbo digitalizacije poslovanja.

- Vodstvu podjetja in vodjem oddelkov nudijo podporo pri poslovnem odločanju.

Podjetja zgoraj naštete prednosti dosežejo z uporabo različnih funkcionalnosti v orodjih MPP, ki omogočajo (Appian, 2018):

- Modeliranje poslovnih procesov: orodja MPP omogočajo grafični prikaz poslovnih procesov v obliki diagramov.
- Upravljanje poslovnih procesov: orodja MPP omogočajo validacijo, analizo in merjenje različnih ključnih kazalnikov uspešnosti v izvedenih poslovnih procesih.
- Oblikovanje poslovnih pravil: orodja MPP omogočajo oblikovanje poslovnih pravil za vsak proces, podproces, aktivnost v procesu itd. v skladu z zakonodajo in predpisi, v okviru katerih mora poslovati podjetje.
- Oblikovanje podatkovnega modela: predstavlja osnovo za razvoj aplikacij v orodjih MPP.
- Izboljšajo sodelovanje in komunikacijo znotraj podjetja: orodja MPP omogočajo preglednejšo izmenjavo mnenj, novih idej in širjenje informacij med udeleženci različnih procesov.
- Integracijo procesov v programske rešitve: orodja MPP omogočajo integracijo procesov v že obstoječe programske rešitve, ki jih imajo v podjetju, npr. MS Sharepoint povezavo z Outlook–om, CRM–jem itd.
- Izdelavo aplikacije, ki podpira točno določen proces v podjetju: naprednejša orodja MPP omogočajo izdelavo specifičnih aplikacij glede na določen poslovni proces v podjetju.
- Vpogled v izvedbo poslovnega procesa v realnem času: orodja MPP omogočajo vpogled v procese v realnem času, izpis poročila o poteku procesov in kreiranje metrike za ugotavljanje uspešnosti ter učinkovitosti izvedbe procesa.

Ko se v podjetju odločajo za nakup orodja MPP kot temeljnega pripomočka pri obvladovanju poslovnih procesov, je priporočljivo, da podjetje skupaj s ponudnikom orodja MPP preuči sedem ključnih točk, ki so pojasnjene v nadaljevanju.

- Cilji, ki jih želi podjetje doseči z uporabo orodja MPP: Podjetje si mora na podlagi jasno opredeljene poslovne strategije in vizije postaviti cilje, ki jih želi doseči z uporabo orodja MPP. Cilji podjetij se med seboj razlikujejo (npr. zmanjšati stroške, zvišati kakovost izvedbe procesov, povečati produktivnost zaposlenih, izvesti integracijo poslovnih procesov v programske rešitve, izvesti prenovo nekaterih poslovnih procesov, izvesti celovito prenovo poslovanja, digitalizirati poslovanje itd.) Odvisno od tega, kaj želi podjetje doseči z uporabo orodja MPP, je odvisna izbira platforme orodja (Aryel Ramos, 2017).
- Izbira ponudnika orodja MPP: Podjetje mora izbrati takšno orodje MPP, ki mu bo služilo kot podpora pri obvladovanju poslovnih procesov. Pri izbiri orodja naj bo podjetje pozorno na število funkcionalnosti, ki jih ponuja orodje MPP (Aryel Ramos,

2017). Smiselno je, da orodje ne vsebuje funkcionalnosti, ki jih v podjetju ne bodo uporabljali oziroma ne bodo doprinesle k uspešnejšemu upravljanju s poslovnimi procesi. Pomembno je, da podjetje izbere takšno orodje MPP, ki bo doprineslo k boljšemu poslovanju in naj ne izbere določenega orodja MPP le zato, ker ga uporabljajo konkurenčna podjetja. Informacije o uporabi orodij MPP konkurenčnih podjetij naj torej podjetje vzame kot primer dobre prakse in ne kot prvo izbiro (Appian, 2018).

- Višina proračuna, namenjenega stroškom nakupa in uporabe orodja MPP: Podjetje mora pred nakupom orodja MPP določiti višino proračuna, ki je povezan z nakupom orodja in ostalimi stroški, ki nastanejo po vpeljavi orodja MPP v podjetje. Pri tem je zelo pomembno, da se določi višina stroškov, ki so posledica integracije programske rešitve v podjetje (npr. stroški usposabljanja zaposlenih pri uporabi nove programske opreme, zakupljenih licenc itd.) V podjetju se morajo zavedati, da bodo šele čez čas vidni tako rezultati uporabe orodja MPP kot tudi povrnitev stroškov, ki so nastali pri naložbi v orodje (Aryel Ramos, 2017).
- Donosnost naložbe v orodje MPP: Poleg stroškov, ki so opredeljeni v prejšnji točki, mora podjetje analizirati, kakšno donosnost bo imelo pri nakupu orodja MPP (Aryel Ramos, 2017). Smiselno je, da podjetje pri potencialnem ponudniku orodja MPP pridobi informacijo o tem, kolikšne so povprečne donosnosti naložb podjetij, ki so uporabljala njihovo orodje MPP. Če bi želelo podjetje dejansko preveriti donosnost naložbe podjetij, bi moral imeti ponudnik orodja MPP študijo primera o uporabi orodja MPP vsakega podjetja, zato naj donosnost naložbe ostalih podjetij upošteva zgolj kot koristno informacijo (Finances Online reviews for business, 2017).
- Uporaba in vzdrževanje orodja MPP v podjetju: Orodje MPP mora biti čim enostavnejše za uporabo, tako za končne uporabnike kot za tiste, ki bodo vzdrževali celotni sistem. Podjetje naj pri ponudnikih orodij MPP pridobi informacijo o zadovoljstvu uporabe orodja MPP pri končnih uporabnikih in kakšne so možne težave pri uporabi ter vzdrževanju orodja (BPMInstitute.org, 2018). Ponudnik orodja MPP naj podjetju predstavi poprodajne aktivnosti v smislu vzdrževalnih pogodb, ki vključujejo vsebinsko in tehnično pomoč končnim uporabnikom.
- Prihodnost uporabe orodja MPP v podjetju: Če želi podjetje doseči čim večjo donosnost orodja MPP, je smiselno, da na dolgi rok velika večina zaposlenih, lastnikov procesov, uporablja to orodje. Na ta način bodo lahko nadzorovali potek poslovnih procesov, merili različne ključne kazalnike uspešnosti (npr. finančne, časovne) poslovnih procesov, integrirali procese v programske rešitve, hitro odkrili ozka grla itd. S pomočjo teh orodij MPP bi medsebojno povezali procese in jih že integrirali v obstoječe programske rešitve (Appian, 2018).
- Možnosti namestitve orodja MPP: Na voljo so različne možnosti uvajanja in namestitve orodja MPP. Podjetje lahko uporabi rešitev v oblaku ali pa ima programsko rešitev shranjeno na lastnih lokalnih strežnikih (BPMInstitute.org, 2018).

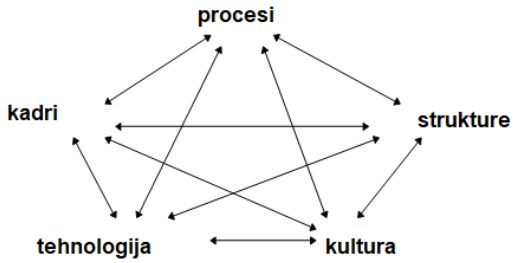
Večina orodij MPP v prihodnosti ne bo služila le za celovito upravljanje poslovnih

procesov (modeliranje, validacijo, simulacijo, optimizacijo in digitalizacijo), ampak se bo uporabnost razširila še na področja poslovne analitike (analiza podatkov v realnem času, velika večina aplikacij bo vključevala rešitve v oblaku, izboljšal se bo pregled nad izvedenimi napakami v samih procesih, hitreje se bo odkrivalo in preprečevalo goljufije strank ter zaposlenih, izboljšale se bodo platforme na mobilnih napravah in tablicah itd.) (BPMInstitute.org, 2018).

1.7 Vidiki prenove poslovnih procesov

Dandanes podjetja menijo, da se prenove poslovanja ne more izpeljati brez napredne informacijske tehnologije, kar vsekakor drži. A uspešne prenove poslovanja ni mogoče doseči zgolj z uporabo napredne informacijske tehnologije, kar je ugotovil tudi ameriški psiholog in profesor dr. Leavitt. S pomočjo socio–tehničnega vidika, ki ga je predstavil v obliki »diamanta«, je želel prikazati, kako je prenova poslovanja v podjetju prepletena tako z družbenimi kot s tehnološkimi dejavniki. Bistvo Leavittovega diamanta je, da vsaka sprememba enega dejavnika v podjetju vpliva na spremembe vseh ostalih dejavnikov. Zato je ob spremembi enega dejavnika v podjetju potrebno prilagoditi in spremeniti tudi vse druge dejavnike. Podjetje ne more doseči uspešne prenove poslovanja, če ne upošteva vseh dejavnikov. Leavittov diamant je prikazan na sliki 4 in je sestavljen iz petih ključnih dejavnikov, ki sestavljajo podjetje: kadri, tehnologija, procesi, strukture in kultura (Kovačič, Jaklič, Indihar Štemberger & Groznik, 2004).

Slika 4: Leavittov diamant



Vir: Kovačič, (1998, str.8).

1.7.1 Sprememba dejavnika »kadri«, če se v podjetju spremenijo ostale štiri komponente

Kadri v podjetju predstavljajo ključni dejavnik, saj brez njih podjetje ne more obstajati. Leavitt zaposlene ne opredeljuje zgolj po delovnih mestih, ki jih zasedajo, npr. revizor, programer, računovodja, analitik itd., temveč poudarja, da se je potrebno osredotočiti na njihova znanja, sposobnosti, produktivnost, spretnosti itd. Pridobiti je potrebno odgovore na naslednja vprašanja: kakšna je njihova motivacija za delo, kakšne so njihove vrednote,

prepričanja, stališča, kakšne nagrade motivirajo zaposlene, da v delo vložijo več truda in kako se bodo odzvali na spremembe v podjetju. Pri prenovi poslovanja imajo prednost tisti zaposleni, ki imajo širše znanje, so produktivnejši, hitreje osvojijo uporabo novih informacijskih rešitev, so fleksibilnejši glede reševanja različnih delovnih nalog itd. (O'Sullivan, 2008, str. 65).

Če se v podjetju spremenijo delovne naloge oziroma opravila, morajo zaposlenim ponuditi izobraževanja in usposabljanja, da bodo lahko samostojno opravljali delo ter prenašali znanje na druge zaposlene.

S prenovo poslovanja se spremeni tudi dejavnik informacijske tehnologije. Še enkrat je potrebno poudariti, da zgolj z uporabo naprednih informacijskih rešitev, ki omogočajo digitalizacijo in integracijo poslovnih procesov, podjetje ne bo doseglo zastavljenih rezultatov, če ne bo upoštevalo tudi ostalih dejavnikov (kadrov, procesov, strukture in kulture) ter jih prilagodilo oziroma spremenilo tako, da bo med vsemi dejavniki čim večja interakcija. Podjetje mora po vpeljavi novih informacijskih rešitev zaposlenim pri njihovi uporabi ponuditi pomoč in izobraževanje (Management.com, 2017).

Pri prenovi poslovanja se spremeni tudi strukturni dejavnik z vidika organiziranosti podjetja. Podjetje mora uvesti takšno organizacijsko strukturo, da bo sodelovanje in komunikacija med zaposlenimi z različnih oddelkov čim večja. Nekateri zaposleni bodo dobili nove vloge, ki bodo zahtevale dodatno izobraževanje in usposabljanje. Smiselno je imenovati skrbnike poslovnih procesov, ki poskrbijo, da se poslovni proces izvede od začetka do konca (Management.com, 2017).

1.7.2 Sprememba dejavnika »tehnologija«, če se v podjetju spremenijo ostale štiri komponente

Tehnologija je dejavnik v podjetju, ki zaposlenim omogoča lažje in hitrejše izvajanje delovnih nalog. Pod tehnologijo štejemo programsko opremo, čitalnike črtnih kod, prenosne računalnike, strojno opremo itd. (O'Sullivan, 2008, str. 65).

Npr. če ima podjetje zaposlene, ki imajo nadpovprečno računalniško pismenost in si tudi v prihodnosti želi zaposlovati kadre, ki so visoko izobraženi in imajo napredno tehnično znanje, potem je primerno, da prilagodijo informacijsko tehnologijo znanju in spretnostim, ki jih imajo zaposleni. Tisti zaposleni, ki niso računalniško pismeni, jih podjetje lahko izobražuje, v nekaterih primerih, ko predstavljajo tehnološki višek, jih lahko tudi odpustijo (Bright hub project management, 2017).

S pomočjo tehnologije se avtomatizirajo določeni deli procesov ali celotni procesi, odpravijo se rutinska dela, kar lahko včasih prinese spremembe v strukturi podjetja. Z organizacijskega vidika se lahko določeni oddelki, v katerih so se opravljale rutinske delovne naloge, ki so po novem avtomatizirane, ukinejo ali združijo z drugimi oddelki.

Lahko nastanejo tudi novi oddelki, kamor se prerazporedijo zaposleni iz ukinjenih oddelkov. Bistveno je, da nova tehnologija predstavlja podporo novi strukturi podjetja (Grover, Jeong Ryul, J.Kettinger & T.C.Teng, 2015).

Spremembe delovnih nalog zahtevajo tudi tehnološko spremembo.

1.7.3 Sprememba dejavnika »struktura«, če se v podjetju spremenijo ostale štiri komponente

Strukturni dejavnik Leavittovega diamanta se ne osredotoča zgolj na hierarhično organiziranost podjetja, ampak vključuje tudi medsebojno sodelovanje med različnimi oddelki in zaposlenimi. Poleg tega preučuje različne nivoje upravljanja podjetja in kako poteka komunikacija ter sodelovanje med različnimi oddelki (O'Sullivan, 2008, str. 66).

Če podjetje zaposluje bolj izobražen kader, ki je bolj usposobljen za reševanje delovnih nalog, je v takem podjetju potrebnega manj nadzora nad zaposlenimi kot v podjetju, ki zaposluje manj usposobljene in manj izobražene kadre. Smiselno je, da se v podjetju nad manj izobraženim in manj usposobljenim kadrom vzpostavi večji nadzor, ki ga izvajajo visoko izobraženi zaposleni (Bright hub project management, 2017).

Vzemimo primer, da se v podjetju zaradi znatno povečanega povpraševanja, ki traja že dve leti, odločijo, da bodo zgradili nov proizvodni obrat, ki bo v drugem kraju kot poteka prvotna proizvodnja. To je velik projekt, saj bo podjetje moralo zgraditi ali pa kupiti nov proizvodni obrat, zaposliti nove delavce, vzpostaviti nadzor delovanja proizvodnje, ki bo deloval na daljavo. Pri tem je pomembno, da se struktura podjetja preoblikuje in prilagodi novemu načinu upravljanja proizvodnje, ki poteka na dveh različnih lokacijah (Bright hub project management, 2017).

Kot smo že večkrat omenili, avtomatizacija poslovanja zahteva tudi prenovo organizacijske strukture. V podjetju lahko ustvarijo nove oddelke in delovna mesta, ki bodo ustreznejša za potrebe poslovanja z novo informacijsko tehnologijo. Nekateri oddelki in delovna mesta se lahko ukinejo, zaposleni pa bodo prerazporejeni na druga delovna mesta ali pa odpuščeni.

1.7.4 Sprememba dejavnika »procesi«, če se v podjetju spremenijo ostale štiri komponente

Procesi so zadnji dejavnik v Leavittovem diamantu. V podjetju se morajo vprašati, kaj želijo doseči z izvajanjem procesov in kakšne koristi imajo od njih, na kakšen način bi jih lahko izboljšali ter kakšno donosnost prinašajo (Dallas & Thandar Wynn, 2014). V začetku tega poglavja je bil poslovni proces že podrobneje opredeljen.

Če se v podjetju odločijo za zaposlovanje kadrov, ki bodo višje izobraženi, bolje

usposobljeni za reševanje delovnih nalog in ki bodo vključeni v izvajanje zahtevnejših poslovnih procesov, potem je smiselno, da ti zaposleni tudi rešujejo težje delovne naloge, ki zahtevajo več znanja in odgovornosti, namesto da opravljajo delovne naloge, ki naj bi jih opravljali zaposleni z nižjo stopnjo izobrazbe (Bright hub project management, 2017).

Če podjetje spreminja svojo strukturo, potem mora temu prilagoditi tudi poslovne procese. Na primer, če so se določeni oddelki v podjetju združili, potem so se spremenili tudi nekateri procesi, ki se po novem izvajajo v združenih oddelkih. Velja tudi obratno, če se en oddelek razdeli na dva ločena oddelka, se s tem spremenijo tudi procesi in delovne naloge (Bednarski, 2014).

Uvedba nove tehnologije zahteva nov način dela. Tako je treba določene procese izpopolniti, dopolniti in jih izvajati bolj uspešno ter učinkovito (Attaran, 2003).

Organizacijska kultura vključuje vse vrednote in prepričanja, ki prispevajo k edinstvenemu družbenemu okolju vsakega podjetja. Je eden izmed najtežjih faktorjev, ki jih je mogoče spremeniti v podjetju, saj vključuje veliko psiholoških dejavnikov. Poleg vrednot in prepričanj vključuje še pričakovanja, izkušnje in filozofijo, ki sestavljajo organizacijo. Izraža se prek načina organizacije dela, obravnavanja zaposlenih in strank podjetja (O'Sullivan, 2008, str. 66). Kaže se tudi v dejstvu, kako so zaposleni predani podjetju in kako delujejo v skladu s prepričanji ter vrednotami podjetja. Organizacijska kultura vpliva na produktivnost zaposlenih in posledično na učinkovitost ter uspešnost poslovanja. Vpletena je v posamezne poslovne procese, kot so proizvodnja (kako in na kakšen način ustvarjati nove produkte), prodaja (kako ravnati s stranko) itd. Opredelitev organizacijske kulture v podjetju je kompleksna naloga, saj temelji na pisanih in tudi nepisanih pravilih (Business Dictionary, 2018).

Prav zato mora vodstvo podjetja pri spremembi vsakega dejavnika iz Leavittovega diamanta ustrezno spremeniti organizacijsko kulturo.

Pri prenovi poslovanja podjetja je zagotovo najtežje spremeniti organizacijsko kulturo, saj je pri tem vključenih mnogo psiholoških dejavnikov. Ob tem pa mora vodstvo na ustrezen način zaposlenim predstaviti vsakršno spremembo, ki se zgodi v podjetju.

2 MODELIRANJE POSLOVNIH PROCESOV

Modeliranje poslovnih procesov je predstavitev poslovnih procesov v obliki procesnih diagramov (Gerth, 2013, str. 15).

Procesni diagram (ang. Business process diagram) je sestavljen iz niza aktivnosti, ki potekajo v določenem zaporedju, njihova izpolnitev pa omogoča, da se dosežejo cilji poslovnega procesa. Je sestavni del analiziranja poslovnih procesov, ki obravnava analizo, načrtovanje, vzdrževanje in izboljšanje poslovnih procesov v podjetju (Gerth, 2013, str.

16).

Zaradi lažjega razumevanja celotnega poslovnega procesa in zaradi napredne informacijske tehnologije se je razvilo več jezikov za modeliranje poslovnih procesov, kot npr. Unified Modeling Language (UML), Business Process Execution Language (BPEL), BPMN itd. (White, 2005).

V magistrskem delu bo v nadaljevanju podrobneje predstavljena BPMN, ki sem jo uporabila v praktičnem delu magistrskega dela za modeliranje poslovnega procesa v programu Bizagi Modeler.

2.1 Grafična notacija BPMN za modeliranje poslovnih procesov

BPMN je grafična notacija, s pomočjo katere grafično ponazorimo poslovni proces, in sicer v obliki procesnega diagrama v aplikaciji, ki je namenjena modeliranju poslovnih procesov. BPMN je standardizirana notacija, ki omogoča zaposlenim oziroma vsem uporabnikom v podjetju enostavno razumevanje procesov. BPMN je leta 2004 razvila skupina Object management group (Gerth, 2013, str. 18).

Danes imamo na voljo več različnih programov, ki omogočajo modeliranje poslovnih procesov v BPMN: JBPM (Eclipse plugin), Signavio, TIBCO Business Studio, IBM Websphere Business Modeler, ARIS, Oracle BPA, Business Process Visual Architect, Progress Savvion Business Modeller, Activiti in mnoge druge (Gerth, 2013, str. 15).

Procesni diagrami, izrisani v BPMN, so sestavljeni iz nabora različnih grafičnih elementov. Njihova uporaba omogoča vsem zaposlenim lažje razumevanje izrisanih poslovnih procesov v obliki procesnih diagramov (White, 2005).

BPMN sestavljajo naslednje štiri osnovne kategorije (Object management group, 2018):

- elementi procesnega toka (ang. Flow objects),
- povezovalni elementi (ang. Connecting objects),
- organizacijski elementi (ang. Swimlanes) in
- artefakti (ang. Artifacts).

2.1.1 Elementi procesnega toka

Osnovni elementi procesnega toka so (Object management group, 2018):

- dogodki (ang. Events),
- aktivnosti (ang. Tasks) in
- odločitvene točke (ang. Gateways).

Vsi osnovni elementi procesnega toka so prikazani na sliki 5.

Slika 5: Dogodek, aktivnost in odločitvena točka

Vir: Object management group (2018).

Dogodki v procesu lahko signalizirajo začetek procesa, konec procesa ali pa se zgodijo med procesom (npr. prejem sporočila, lahko signalizirajo, da je prišlo do napake, da bo prišlo do zamude itd.). Vmesnih dogodkov ne uporabljamo na začetku in na koncu procesa. Na sliki 6 so prikazani trije osnovni dogodki, ki nakazujejo začetek procesa, konec procesa in dogodke, ki se zgodijo med procesom. Seveda obstaja več vrst začetnih dogodkov, končnih dogodkov in vmesnih dogodkov (Freund & Rucker, 2012, str. 34).

Slika 6: Osnovni dogodki – začetni dogodek, končni dogodek in vmesni dogodek

Vir: Object management group (2018).

Aktivnosti v procesu predstavljajo neko delo, ki se izvede med procesom. Poznamo različne vrste aktivnosti (npr. prejemanje in pošiljanje), uporabimo jih glede na delovne naloge v procesu. Če ne uporabimo nobene aktivnosti, ki so na voljo v BPMN, takšno aktivnost imenujemo slepa aktivnost. Posebna vrsta aktivnosti je aktivnost, ki predstavlja podproces. Tako kot obstaja več vrst dogodkov, poznamo tudi več vrst aktivnosti (Freund & Rucker, 2012, str. 17).

Na sliki 7 je prikazana slepa aktivnost in aktivnost, ki predstavlja podproces.

Slika 7: Slepa aktivnost in aktivnost, ki predstavlja podproces

Vir: Object management group (2018).

Prehodi v procesu lahko predstavljajo odločitveno točko. Uporabljamo jih tudi za razvejanje in združevanje poti. Poznamo jih več vrst, uporabimo jih glede na tok procesa (Freund & Rucker, 2012, str. 18–26) . Na sliki 8 je prikazan osnovni gradnik prehodov.

Slika 8: Osnovna odločitvena točka

Vir: Object management group (2018).

2.1.2 Povezovalni elementi

Med povezovalne elemente spadajo naslednje osnovne povezave (Object management group, 2018):

- tok zaporedja (ang. Sequence flow),
- tok komunikacije (ang. Message flow) in
- asociacijska povezava (ang. Association).

Tok zaporedja, ki je prikazan na sliki 9, je ponazorjen s polno črto in puščico. V poslovnem procesu nakazuje zaporedni tok aktivnosti v procesu (Weske, 2007, str. 210).

Slika 9: Tok zaporedja

Vir: Object management group (2018).

Tok komunikacije, ki je prikazan na sliki 10 ponazarja pretok sporočil. Povezava je ponazorjena s črtkano črto, praznim krogom na začetku povezave in prazno puščico na koncu povezave. Prikazuje, katera sporočila potekajo med organizacijskimi objekti (bazeni in proge). Te povezave nikoli ne moremo uporabiti znotraj enega bazena za povezovanje dogodkov, aktivnosti in odločitvenih točk (Weske, 2007, str. 211–212).

Slika 10: Tok komunikacije

Vir: Object management group (2018).

Asociacijska povezava, ki je prikazana na sliki 11, je ponazorjena s pikčasto črto. Uporablja se za povezovanje elementov procesnega toka (aktivnosti, dogodkov in odločitvenih točk) z artefakti (podatkovnimi objekti in opombami).

Uporaba te povezave v procesnem diagramu ne vpliva na potek poslovnega procesa (Weske, 2007, str. 209).

Slika 11: Asociacijska povezava

Vir: Object management group (2018).

2.1.3 Organizacijski objekti

Osnovna organizacijska objekta sta (Object management group, 2018):

- bazen (ang. Pool) in
- proga (ang. Line).

Bazen, ki je prikazan na sliki 12, se uporablja za združevanje aktivnosti določenega procesa. Znotraj bazena med seboj komunicirajo različni subjekti. Bazen lahko vsebuje več prog (Freund & Rucker, 2012, str. 95).

Slika 12: Bazen

Vir: Object management group (2018).

Proge ločujejo aktivnosti med udeleženci v procesu. Po njih poteka komunikacija med subjekti (Freund & Rucker, 2012, str. 95). Primer proge je prikazan na sliki 13.

Slika 13: Proga

Vir: Object management group (2018).

2.1.4 Artefakti

Artefakti uporabniku poslovnega procesa ponujajo dodatne, podrobnejše informacije o procesu, določeni aktivnosti, dogodku itd. Njihova uporaba v diagramu procesa ne vpliva na sam potek procesa (Freund & Rucker, 2012, str. 78).

Osnovni elementi artefaktov so naslednji (Objet management grup, 2018):

- podatkovni objekti (ang. Data object),

– skupina (ang. Group) in
– opomba (ang. Annotation).

Podatkovni objekt, ki je prikazan na sliki 14, se uporablja za prikaz uporabnikom, kateri podatki so potrebni za izvedbo določene aktivnosti oziroma katere podatke dobimo pri izvedbi določene aktivnosti. Podatkovni objekti ne vplivajo na sam potek poslovnega procesa. Poznamo več vrst podatkovnih objektov (Freund & Rucker, 2012, str. 78–79).

Slika 14: Podatkovni objekt

Vir: Object management group (2018).

Skupina, ki je prikazana na sliki 15, se uporablja za združevanje različnih aktivnosti v procesu in ne vpliva na sam potek procesa. Aktivnosti lahko združujemo iz različnih bazenov (Freund & Rucker, 2012, str. 78–79).

Slika 15: Skupina

Vir: Object management group (2018).

Opomba, ki je prikazana na sliki 16, se uporablja za podrobnejši opis posamezne aktivnosti, prehoda, odločitvene točke, povezave itd. v procesnem diagramu. Ne vpliva na potek poslovnega procesa, je pa v pomoč različnim uporabnikom poslovnega procesa, saj omogoča boljše razumevanje (Freund & Rucker, 2012, str. 79).

Slika 16: Opomba

Vir: Object management group (2018).

2.2 Primer modela poslovnega procesa izrisanega v BPMN

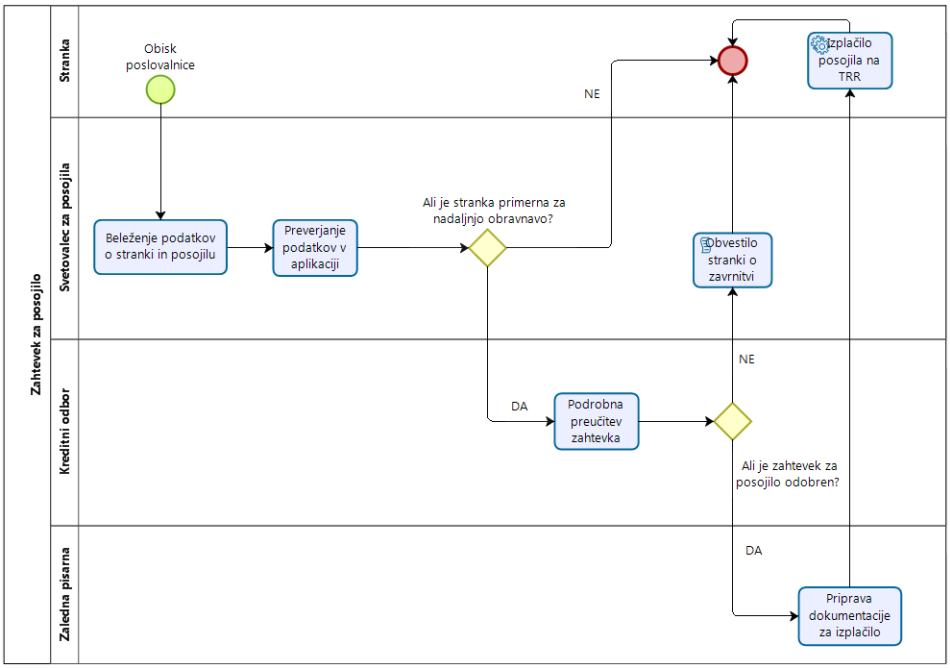
V nadaljevanju je opisan proces, kjer so uporabljeni osnovni elementi procesnega toka. Izbran je primer procesa, ki poteka v bankah pri odobritvi posojil.

Poslovni proces je izrisan v programu Bizagi Modeler. Izbrani model poslovnega procesa bi lahko izrisala v kateremkoli drugem orodju MPP, ki uporablja BPMN za izris poslovnih procesov.

Ko se stranka odloči za najem posojila, obišče bančno poslovalnico. V poslovalnici ji svetovalec za posojila naredi informativne izračune različnih možnosti posojil in zbere vso potrebno dokumentacijo o stranki (npr. številko osebnega dokumenta, davčno številko, plačilne liste stranke itd.). Vse potrebne podatke svetovalec vpiše v aplikacijo, ki je namenjena oblikovanju zahtevkov za posojilo. Na podlagi finančnega stanja stranke aplikacija naredi osnovni izračun, ali je zahtevek za posojilo sploh primeren za nadaljnjo obravnavo. Če po izračunih aplikacije zahtevek ni primeren, se proces zaključi že v poslovalnici, saj stranka ni kreditno sposobna. Če pa je zahtevek za posojilo na podlagi preverjanja podatkov v aplikaciji primeren za nadaljnjo obravnavo, potem o odobritvi kredita ne odloča več svetovalec v poslovalnici, temveč gre zahtevek v obravnavo na višji nivo, na kreditni odbor. Kreditni odbor sestavlja več finančnih analitikov, ki pregledajo in ocenijo kreditno sposobnost strank. V primeru, da kreditni odbor odloči, da stranka ni kreditno sposobna, jo o tem obvesti pisno prek elektronske pošte. Če kreditni odbor zahtevek odobri, potem preda vso potrebno dokumentacijo v zaledno pisarno. Tam zaposleni referenti, za stranko pripravijo vso potrebno dokumentacijo za izplačilo kredita in poskrbijo, da se stranki posojilo izplača na transakcijski račun. Z izplačilom posojila kreditno sposobni stranki se proces zaključi (Bizagi Modeler, 2018).

Celotni proces je prikazan na sliki 17.

Slika 17: Osnovni proces odobritve posojila v bankah



Vir: Povzeto in prirejeno po Bizagi, (2018).

Vsak proces ima svoje ime, v našem primeru je ime procesa Zahtevek za posojilo. V procesu imamo štiri udeležence, in sicer stranko, svetovalca za posojila, kreditni odbor ter zaledno pisarno.

V procesnem diagramu imamo naslednje BPMN gradnike (Bizagi, 2018):

- Dva dogodka, ki ponazarjata bodisi začetek, bodisi konec procesa.
- Šest različnih vrst aktivnosti, ki ponazarjajo določeno aktivnost v procesu.
- Dva osnovna prehoda, ki predstavljata odločitveno točko. Glede na izbiro, ki jo nudi odločitvena točka, je odvisen nadaljnji potek procesa.

Zgoraj opisani proces je v resnici veliko bolj kompleksen. Že pri samem začetku procesa bi lahko stranko opredelili kot fizično ali pravno osebo, v nadaljevanju bi ji glede na to dodelili svetovalca za posojila, bodisi za fizične ali za pravne osebe. Glede na to, da nekatere vrste kreditov (npr. manjše potrošniške kredite) lahko odobri oziroma zavrne že svetovalec v poslovalnici, bi bilo smiselno podrobneje opredeliti za katero vrsto posojila se zanima stranka itd.

Iz konkretnega primera je razvidno, da je že sam popis poslovnega procesa zelo pomemben. Tisti, ki popisuje poslovne procese, mora dobro poznati celotno sliko izvedbe procesa, saj bo na podlagi tega lahko izrisal procesni diagram, ki bo predstavljal podrobno razdelani proces in ki bo primeren za nadaljnjo obravnavo (tj. analizo, optimizacijo in digitalizacijo).

2.3 Bizagi BPM Suite

Bizagi je zasebno podjetje, ki je bilo ustanovljeno leta 1989, s sedežem v Združenem kraljestvu. Podjetje razvija programsko opremo, ki je namenjena podpori managementu poslovnih procesov (Bizagi, 2018).

Bizagi BPM Suite zajema tri produkte (Bizagi, 2018): Bizagi Modeler, Bizagi Studio in Bizagi Engine. Na sliki 18 so prikazani vsi logotipi programskega paketa Bizagi BPM Suite.

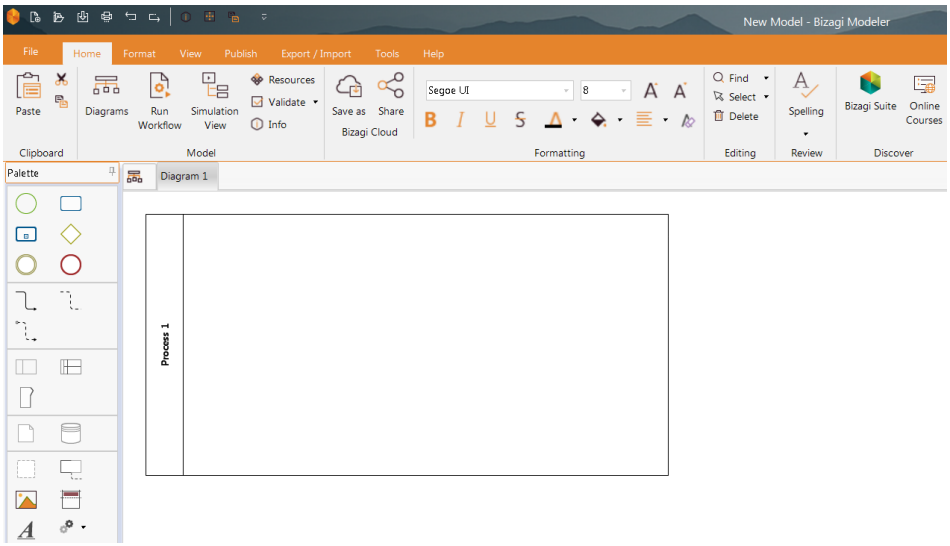
Slika 18: Bizagi BPM Suite: Bizagi Modeler, Bizagi Studio, Bizagi Engine.

Vir: Bizagi (2018).

Bizagi Modeler in Bizagi Studio sta prosto dostopna na spletu. Bizagi Engine, ki omogoča uporabo aplikacije več kot tisoč uporabnikom v podjetjih, pa je plačljiv. Vsi trije produkti so na voljo uporabnikom v več jezikih (angleščini, japonščini, nemščini, italijanščini, ruščini, portugalščini, francoščini, kitajščini, španščini in nizozemščini). Uporabnikom je pri uporabi Bizagijevih produktov na voljo pomoč na uradni spletni strani podjetja Bizagi v obliki predstavitvenih filmčkov, na forumih itd. (Bizagi, 2018).

Bizagi Modeler je aplikacija, ki omogoča izris diagrama poslovnega procesa v BPMN, omogoča simulacijo procesov in dokumentiranje procesov. Procesi se lahko objavijo v Wordu, PDF–ju, SharePointu, na Wiki–ju in na različnih spletnih straneh. Mogoče jih je shraniti v različnih slikovnih formatih (.png, .jpg, .svg) (Bizagi Modeler, 2018). Na sliki 19 je prikazan Bizagi Modeler v stanju, ko uporabnik odpre aplikacijo.

Slika 19: Bizagi Modeler



Vir: Bizagi Modeler (2018).

V praktičnem delu magistrske naloge je predstavljena analiza obstoječega in prenovljenega procesa s pomočjo naslednjih stopenj, ki so na voljo v programu Bizagi Modeler (Bizagi Modeler, 2018):

- validacija procesa (ang. Process Validation),
- časovna analiza (ang. Time Analysis),
- analiza virov (ang. Resource Analysis) in
- analiza koledarja (ang. Calendar Analysis).

Navedene stopnje so podrobneje opisane v nadaljevanju.

Validacija procesa je prva stopnja analize poslovnega procesa. Uspešno izvedena validacija zagotavlja, da proces poteka skozi vse zaporedne tokove tekom procesa in se izvaja v skladu s pričakovanji. Z validacijo preverimo in dokažemo, da so vsi prehodi ter vsa sporočila sinhronizirani in da je verjetnost odločitev pravilno oziroma smiselno dodeljena (Bizagi Modeler, 2018).

Za izvedbo validacije je potrebno nastaviti število iteracij izvajanja celotnega procesa in verjetnost odločitev pri prehodih. Če želimo dobiti zanesljive in stabilne rezultate, je potrebno nastaviti zelo veliko število iteracij procesa. Priporočljivo je namreč nastaviti vsaj tisoč iteracij. Če je validacija uspešno izvedena, bo rezultat validacije število vseh iteracij procesa, ki želimo, da se izvedejo. Povedano drugače, vsak začetek procesa ima svoj konec – število začetnih dogodkov se ujema s številom vseh iteracij, ki so bile izvedene (Bizagi Modeler, 2018).

Časovna analiza je druga stopnja analize poslovnega procesa. Uporablja se pri merjenju časa izvedbe celotnega procesa (od začetka do konca), časa trajanja izvedbe določene aktivnosti v več izvedenih iteracijah itd. Pri tem privzamemo, da imamo neskončno virov oziroma da ne definiramo vrednosti za spremenljivke virov. Če želimo izmeriti celotno trajanje procesa, moramo vsaki aktivnosti v procesu določiti čas trajanja (Bizagi Modeler, 2018).

Analiza virov je tretja stopnja analize procesa in prikaže rezultate izvedbe procesa glede na omejitve virov. Vir v poslovnem procesu je opredeljen kot oseba, prostor, finančni okvir, oprema itd. in je pomemben pri izvedbi določene naloge med procesom. Na drugi stopnji simulacije (analiza časa) privzeto simuliramo proces za neskončno virov, kar v realnem poslovnem svetu ni smiselno, zato na tretji stopnji podrobneje simuliramo poslovni proces glede na dane vire. Rezultati analize virov so zelo koristni, saj lahko prek rezultatov dobimo naslednje podatke: koliko znašajo celotni stroški virov, ki so vključeni v proces, ali so viri premalo ali prekomerno izkoriščeni in kolikšen je pričakovani čas izvedbe procesa glede na trenutno razporeditev virov. Na tretji stopnji dobimo tudi rezultate časovne analize (Bizagi Modeler, 2018).

Analiza koledarja je četrta in hkrati zadnja stopnja analize procesa. Na tej stopnji

simuliramo razpoložljivost virov v dinamičnih časovnih obdobjih. Upoštevamo tudi praznike, vikende, odmore, delovni čas zaposlenih. Z rezultati, ki jih pridobimo, lahko dejansko ocenimo učinkovitost procesa, ki se odvija v realnih okoliščinah v poslovnem svetu. Na koncu te stopnje pridobimo natančnejše informacije o celotnih stroških izvedbe procesa, pričakovanem času izvedbe procesa glede na podane časovne omejitve (odmori zaposlenih, vikendi in prazniki) in informacije o izkoriščenosti različnih virov. Na podlagi tega lahko ocenimo učinkovitost procesa (Bizagi Modeler, 2018).

Za pridobitev rezultatov simulacije analize koledarja kot vhodne podatke potrebujemo vse tiste, ki so bili uporabljeni za izvedbo tretje stopnje, ter dodamo spremenljivke, ki opredeljujejo urnike zaposlenih, praznike, vikende itd. S tem dobimo realne rezultate izvedbe poslovnega procesa v resničnem poslovnem okolju (Bizagi Modeler, 2018).

Bizagi Studio je aplikacija, ki omogoča podjetjem, da na enostaven način digitalizirajo poslovne procese s pomočjo aplikacij, narejenih v Bizagi Studio. Poslovni proces, izrisan v programu Bizagi Modeler, namreč pretvorijo v aplikacijo, ki je prilagojena najnovejšim spletnim standardom. Bizagi Studio omogoča izdelavo grafičnega vmesnika, kreiranje podatkovnega modela, poslovnih pravil, obrazcev itd. Tako kot program Bizagi Modeler omogoča izris diagrama poslovnega procesa v BPMN, omogoča tudi simulacijo procesov in dokumentiranje procesa. S pomočjo izrisanega procesa pa uporabnik ustvari aplikacijo, v katero je implementiran proces, ki ga je izrisal v obliki diagrama. Izdelava aplikacije poteka po principu povleci in spusti oziroma angleško »drag and drop« (Bizagi Studio, 2018).

Aplikacije izdelane v Bizagi Studio omogočajo nastavitev koledarja, optimalno razporeditev virov med izvajanjem procesov, popoln nadzor nad izvedenimi aktivnostmi in viri. Možna pa je tudi povezava z aplikacijami kot so Outlook, SharePoint, z različnimi moduli v SAP–u itd. (Bizagi Studio, 2018).

Bizagi Engine. Ko imajo v podjetju aplikacije izdelane v Bizagi Studio, lahko začnejo z uporabo Bizagi Engine. To je orodje, ki omogoča izvajanje aplikacij in njihov prenos na namizne računalnike, prenosne računalnike, tablice ter mobilne telefone. Uporablja jih lahko več kot tisoč uporabnikov v podjetjih, organizacijah, javnih zavodih itd. Deluje na način zakupa uporabniških licenc. Orodje zagotavlja, da vsak korak v procesu poteka točno tako kot je zastavljen. Omogočen je vpogled v izvajanje procesa v realnem času. Z grafičnimi prikazi omogoča sledenje in spremljanje dela vseh zaposlenih, ki so udeleženi v proces (Bizagi Engine, 2018).

3 PROCES TESTIRANJA PROGRAMSKIH REŠITEV V IZBRANEM PODJETJU

Izbrano podjetje posluje kot družba z omejeno odgovornostjo in ima več kot 30 zaposlenih. Je ponudnik programskih rešitev s področja kvantitativnih financ in napredne analitike za

finančne institucije. Ukvarja se z razvojem, implementacijo, nadgradnjo in vzdrževanjem programske opreme ter svetovanjem podjetjem pri izbiri ustreznega produkta, prilagojenega njihovemu poslovanju. Organizacijska struktura podjetja se deli na pet oddelkov: razvojni oddelek, tehnološki oddelek, oddelek za pomoč uporabnikom, kadrovska služba in prodaja. Strokovnjaki, ki so zaposleni v podjetju, imajo znanja s področja financ in računalništva. Vsebinsko in tehnično poznavanje področij sta ključna dejavnika za uspešno zaključene projekte. Programske rešitve izbranega podjetja pripomorejo k dvigu uspešnosti in učinkovitosti poslovanja podjetij, upravljanje njihovih tveganj ter portfelja naložb in k uvedbi ter uporabi poslovne inteligence v njihovo poslovno okolje (Izbrano podjetje, 2018).

V izbranem podjetju svetujejo, razvijajo in nadgrajujejo programske rešitve, ki finančnim institucijam nudijo podporo pri (Izbrano podjetje, 2018):

- upravljanju portfelja finančnih naložb;
- knjiženju in vrednotenju naložb po najnovejših mednarodnih računovodskih standardih;
- izračunu denarnih tokov;
- modeliranju in avtomatskem merjenju obrestnega, likvidnega, tržnega in kreditnega tveganja v skladu z zakonskimi zahtevami (npr. Basel II);
- avtomatičnem zbiranju in shranjevanju finančnih podatkov, npr. obrestnih mer, podatkov o dividendah in kuponih obveznic itd.;
- načrtovanju in implementaciji podatkovnih skladišč na vseh vodilnih platformah (IBM, Oracle, Microsoft);
- razvoju in nadgradnjah programskih rešitev s področja poslovne analitike, prediktivne analitike ter multimedijske analize;
- implementaciji poslovne inteligence v njihovo poslovanje.

V izbranem podjetju so zaradi povečanega povpraševanja podjetij po njihovih produktih v zadnjem letu več kot podvojili število novih sodelavcev. Ker se je v kratkem času močno povečalo število zaposlenih, so se v podjetju odločili, da bodo temeljne poslovne procese, ki so vključeni v življenjski cikel razvoja programskih rešitev, podrobneje razdelali, preučili, prenovili in določene procese deloma implementirali v aplikacije.

Izbranemu podjetju sem predlagala konkretne rešitve pri prenovi in digitalizaciji procesa testiranja programskih rešitev. Obstoječi proces testiranja je bil namreč zelo težko obvladljiv in ni bil prilagojen trenutni organizacijski strukturi ter poslovanju podjetja. Kot omenjeno, je podjetje močno povečalo število zaposlenih, tudi število testerjev je v enem letu naraslo s pet na trinajst. Ker se je v zelo kratkem času precej povečalo število zaposlenih testerjev in ker v izbranem podjetju razvijajo in nadgrajujejo vsebinsko zahtevne aplikacije, obstoječi proces testiranja ni bil več ustrezen.

Zaposleni, ki so vključeni v obstoječi proces testiranja programskih rešitev, ga težko nadzorujejo, saj nimajo aplikacije, ki bi podpirala obstoječi proces. Novo zaposleni testerji

niti dobro ne vedo, kako poteka celoten proces, kakšna je njihova vloga v celotnem procesu in katere so njihove delovne naloge. Zato sem se odločila, da s pomočjo orodja MPP – Bizagi Modeler in Bizagi Studio – naredim obsežnejšo analizo obstoječega in predlaganega prenovljenega procesa ter ustvarim aplikacijo, ki bo podpirala prenovljeni proces testiranja in bo v pomoč zaposlenim, ki so vanj vključeni.

3.1 Temeljni in podporni procesi v izbranem podjetju po Porterjevi verigi vrednosti

Opredelitev temeljnih in podpornih poslovnih procesov, ki se odvijajo v izbranem podjetju, bom predstavila v obliki Porterjeve verige vrednosti. Kot je znano, je Porter oblikoval verigo vrednosti na osnovi proizvodnega podjetja (Porter, 1998, str. 36). Zasnoval jo je tako, da jo lahko uporabi prav vsako podjetje, ne glede na to, kakšna je njegova primarna dejavnost.

Temeljni procesi v izbranem podjetju po Porterju so tisti, ki neposredno ustvarjajo dodano vrednost za naročnika programskih rešitev in izbrano podjetje. Temeljni poslovni procesi v izbranem podjetju so naslednji: proces razvoja programskih rešitev, proces testiranja programskih rešitev, proces pomoč uporabnikom in prodajni proces. Vsi temeljni poslovni procesi so podrobneje opisani v nadaljevanju.

Proces razvoja programskih rešitev: v ta proces so vključeni razvijalci, ki razvijajo in testirajo programske rešitve. Proces poteka v razvojnem oddelku podjetja. Izhod omenjenega procesa je izdelana programska rešitev za naročnika, ki je pripravljena na obsežnejše testiranje v tehnološkem oddelku (Izbrano podjetje, 2018b).

Proces testiranja programskih rešitev: v proces so vključeni testerji, ki izvedejo alfa testiranje. Proces poteka v tehnološkem oddelku podjetja. Izhod omenjenega procesa je kakovostna programska rešitev, katere funkcionalnosti delujejo brezhibno in zagotavljajo nemoten potek dela pri naročniku (Izbrano podjetje, 2018b).

Proces pomoč uporabnikom: v proces so vključeni strokovni sodelavci za pomoč uporabnikom, ki nudijo vsebinsko in tehnično pomoč končnim uporabnikom aplikacij. Proces poteka v oddelku pomoč uporabnikom. Proces spada med temeljne procese, saj podjetje z njim tudi po prodaji programske rešitve še vedno ustvarja prihodke v obliki vzdrževalnih pogodb in gradi odnos z uporabnikom programske rešitve. Izhod omenjenega procesa so rešeni zahtevki uporabnikov in izdani računi naročniku, ki izbranemu podjetju omogočajo ustvarjanje prihodkov in večanje dobička (Izbrano podjetje, 2018b).

Proces prodaje: v proces je vključen prodajni predstavnik, ki je zadolžen za prodajo programskih rešitev, sklepanje prodajnih pogodb z naročniki in za sklepanje pogodb o poprodajnih aktivnostih. Proces prodaje omogoča tudi spoznavanje stranke, kar je zelo pomembno v procesu pomoči uporabnikom. Izhod prodajnega procesa je prodana

programska rešitev in sklenjena vzdrževalna pogodba, ki izbranemu podjetju omogoča ustvarjanje prihodkov in večanje dobička (Izbrano podjetje, 2018b).

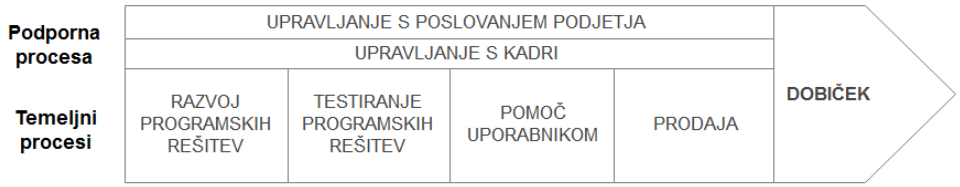
Podporni procesi po Porterjevi verigi vrednosti so tisti, ki omogočajo, da se uspešno izvedejo temeljni poslovni procesi. V izbranem podjetju sta dva podporna procesa, in sicer upravljanje s kadri in upravljanje s poslovanjem podjetja.

Proces upravljanja s kadri: v proces je vključen kadrovik, ki je zadolžen, da izvede celotni zaposlitveni proces novega zaposlenega v podjetju (od objave razpisa za delovno mesto do priprave pogodbe za novo zaposlenega). V procesu upravljanja s kadri mora opravljati tudi naslednje delovne naloge: organizirati zdravniške preglede za zaposlene, pripravljati podatke za obračun plač zaposlenih za zunanje računovodstvo, prijaviti izbrano podjetje na javne razpise, organizirati različne oblike izobraževanj za zaposlene itd. Kadrovik tesno sodeluje z vodstvom podjetja pri izboljšanju procesa upravljanja s kadri. Glavni izhod procesa upravljanja s kadri je zaposlitev novega zaposlenega in urejena kadrovska mapa posameznega zaposlenega (Izbrano podjetje, 2018b).

Upravljanje s poslovanjem podjetja: v proces je vključen lastnik podjetja, ki je tudi direktor podjetja in ostali solastniki podjetja, ki odločajo, na katere javne razpise se bo podjetje prijavilo, katera podjetja, ki so lahko bodoči naročniki programskih rešitev, naj obišče prodajni predstavnik; kakšna naj bo organizacija oddelkov, da bo poslovanje podjetja čim boljše; nenehno spremljajo finančne rezultate poslovanja; skupaj s kadrovsko službo določijo, katere kadre se bo zaposlovalo; kakšne bodo finančne nagrade za zaposlene in na kakšen način najti primerne tehnične kadre na trgu dela. Vodstvo podjetja nadzira izvajanje vseh procesov v podjetju, tako temeljnih kot podpornih (Izbrano podjetje, 2018b).

Na sliki 20 je prikazana zgoraj opisana Porterjeva veriga vrednosti s temeljnimi in podpornimi procesi v izbranem podjetju.

Slika 20: Porterjeva veriga vrednosti s temeljnimi in podpornimi procesi v izbranem podjetju



Vir: lastno delo.

3.2 Življenjski cikel razvoja programske opreme v izbranem podjetju

Ker je proces testiranja vključen v življenjski cikel razvoja programske opreme, sem v tem razdelku opisala celotni življenjski cikel razvoja programske opreme v izbranem podjetju.

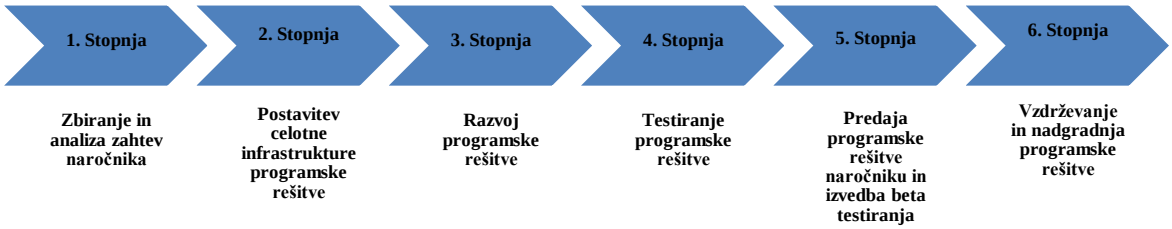
V izbranem podjetju so za osnovni proces razvoja novih funkcionalnosti v programskih rešitvah vzeli splošni proces razvoja programskih rešitev. Proces razvoja programske rešitve oziroma življenjski cikel razvoja programske rešitve (ang. Software development life cycle for software development, v nadaljevanju SDLC) je proces, katerega rezultat je kakovostna programska rešitev za naročnika. SDLC uporabljajo pretežno podjetja, katerih primarna dejavnost je razvoj in nadgradnja programskih rešitev (Ruparelia, 2010). V življenjskem ciklu razvoja programske rešitve nastopa šest stopenj. Vsaka stopnja predstavlja podproces v celotnem procesu razvoja programske rešitve in je pomembna pri njenem nastajanju (Stackify, 2018). Pomanjkljiva oziroma nepravilna izvedba ene same stopnje oziroma podprocesa lahko povzroči resne težave pri nastajanju programske rešitve in zmanjša kakovost končnega izdelka. Povzroči lahko tudi dodatne stroške za naročnika programske rešitve in izgubo prihodkov za podjetje, ki programsko rešitev ustvarja (Izbrano podjetje, 2016a).

Obstaja več SDLC modelov kot npr. model slapa (ang. Waterfall model), V–Model, spiralni model (ang. Spiral model) itd. Skupno vsem procesnim modelom je, da vsebujejo vseh šest osnovnih stopenj življenjskega cikla razvoja programske rešitve (Ruparelia, 2010).

V izbranem podjetju imajo že razvite programske rešitve, ki jih prodajajo na trgu. Trenutno razvijajo in nadgrajujejo obstoječe programske rešitve, ki so odraz naročnikovega povpraševanja. Naročniki programskih rešitev od izbranega podjetja kupijo osnovno verzijo programske rešitve, ki vsebuje osnovne funkcionalnosti. Glede na želje in potrebe naročnika v izbranem podjetju nadgradijo programsko rešitev prek razvoja novih funkcionalnosti.

Kot že omenjeno, v izbranem podjetju uporabljajo za nadgradnjo programskih rešitev klasičen šeststopenjski model SDLC, ki je sestavljen iz šestih zaporednih stopenj. Vseh šest stopenj procesa (podprocesov), ki so predstavljene na sliki 21, je opisanih v nadaljevanju.

Slika 21: Življenjski cikel razvoja programske rešitve v izbranem podjetju



Vir: Izbrano podjetje (2016a).

– Zbiranje in analiza zahtev naročnika

V kateremkoli SDLC modelu je prva stopnja hkrati tudi glavna faza. V tej fazi namreč poteka iskanje idej in rešitev (ang. brainstorming) za razvoj novih funkcionalnosti. Pomembno je, da naročnik programske rešitve natančno razume poslovni problem in da zna definirati zahteve po novi oziroma nadgrajeni programski rešitvi. Zaposleni v podjetju, kjer razvijajo programsko rešitev (projektni vodja, arhitekt programske rešitve in razvijalec programske rešitve), morajo razumeti poslovni problem naročnika, saj mu le tako lahko svetujejo, kakšna programska rešitev bi bila najprimernejša za njegovo poslovanje (Stackify, 2018). V prvi fazi podjetje, ki prodaja programsko rešitev, in naročnik programskih rešitev določita časovni in stroškovni okvir razvoja programske rešitve. Projektni vodja skupaj z razvijalci oblikuje specifikacijo, v kateri so natančno opisani poslovni problem in zahteve naročnika (Ghahrai, 2018). Podjetje, ki razvija programske rešitve, naročniku pošlje ponudbo, v kateri so specificirani vsi stroški, povezani z razvojem programske rešitve in opisane funkcionalnosti, ki jih programska rešitev vsebuje (Izbrano podjetje, 2016a).

Priporočljivo je, da se na strani podjetja, ki je naročnik programske rešitve, čim več zaposlenih udeleži sestankov s predstavniki podjetja, ki bodo sodelovali pri razvoju programske rešitve. Le tako si lahko obe strani izmenjata čim več informacij v zvezi s poslovnimi potrebami naročnika in o dejanskih možnostih razvoja programske rešitve (Alwan, 2015).

– Postavitev celotne infrastrukture programske rešitve

Na drugi stopnji arhitekt programske rešitve in vodilni razvijalci programske rešitve sestavijo tehnično specifikacijo, ki predstavlja načrt za razvoj programske rešitve (Ghahrai, 2018). V tehnični specifikaciji natančno določijo kakšna naj bo sistemska zasnova aplikacije in postavijo zahteve za oblikovanje strojne opreme ter varnostnega sistema aplikacije (Alwan, 2015).

Obenem pa je sistemska zasnova podlaga za oblikovanje celotne arhitekturne rešitve programske rešitve, ustvari se načrt glede baze podatkov (v koliko tabel se bodo shranjevali podatki iz aplikacije, kako bodo tabele med seboj povezane, kako bo izgledal grafični vmesnik itd.) (Izbrano podjetje, 2016a).

Na tej stopnji testerji programskih rešitev skupaj z razvijalci oblikujejo proces testiranja programske rešitve. Vsi, ki so vključeni v življenjski cikel razvoja programske rešitve (razvijalci in testerji programskih rešitev, projektni vodja in drugi strokovnjaki, ki sodelujejo pri nastajanju programske rešitve) na tej stopnji dobijo podroben načrt izvedbe celotnega življenjskega cikla razvoja programske rešitve.Tako je celotni proces razvoja programske rešitve bolj obvladljiv in razumljiv za vse udeležence, ki nastopajo v procesu razvoja programske rešitve (Ghahrai, 2018).

– Razvijanje programske rešitve

Tretja stopnja je v celotnem življenjskem ciklu razvoja programske rešitve najdražja in časovno najdaljša. Razvijalci dobijo od arhitekta programske rešitve natančna navodila, kako naj razvijejo kodo za programsko rešitev. Pomembno je, da razvijalci dobro razumejo poslovni problem, saj bodo le tako lahko napisali ustrezno kodo programa, ki bo zadoščala zahtevam naročnika (Alwan, 2015). Običajno obsežnejšo programsko rešitev razvija več razvijalcev, v primeru dodajanja enostavnejših funkcionalnosti pa po en razvijalec razvija kodo za določeno funkcionalnost . Tudi v izbranem podjetju navadno en razvijalec razvija določeno funkcionalnost (Izbrano podjetje, 2015). V fazi razvoja programske rešitve se izvede tudi pregled kode programske rešitve (ang. code review), za katerega je zadolžen izključno izkušen razvijalec (ang. senior developer). Izkušen razvijalec namreč preveri pravilnost delovanja kode v smislu, ali so vključene vse potrebne funkcionalnosti, ki jih je zahteval naročnik, pa tudi, če so pravilno uporabljene in napisane metode znotraj razredov itd. (Ghahrai, 2018) Če izkušen razvijalec ob pregledu kode ni našel napak oziroma pomanjkljivosti v kodi, gre lahko koda na naslednjo stopnjo, ki je testiranje programske rešitve. V nasprotnem primeru pa mora razvijalec, ki je napisal kodo programske rešitve, odpraviti napake v kodi (Izbrano podjetje, 2015). V sklopu tretje stopnje naj bi razvijalec, ki je razvijal kodo, izvedel osnovni test funkcionalnosti pred obsežnejšo stopnjo testiranja programske rešitve (Ghahrai, 2018).

V izbranem podjetju je v proces testiranja vključen tudi pregled kode in osnovni test, ki ga izvede razvijalec, ki je razvijal kodo (Izbrano podjetje, 2016a).

– Testiranje programske rešitve

Na tej stopnji nastopi testiranje programskih rešitev, ki je tudi obravnavani proces v magistrskem delu. Testerji programskih rešitev (v nadaljevanju testerji), so zadolženi, da preverijo, ali vse funkcionalnosti v programski rešitvi delujejo pravilno. V ta namen si pripravijo nabor testnih scenarijev s pričakovanimi rezultati in z njimi preverijo delovanje funkcionalnosti (Ghahrai, 2018). Pomembno je, da so testni scenariji sestavljeni premišljeno in da tester čim bolje razume delovanje funkcionalnosti, saj bo le v tem primeru lahko celovito testiral funkcionalnost in našel napake oziroma prepoznal pomanjkljivosti v delovanju funkcionalnosti (Izbrano podjetje, 2016a). Stopnja testiranja se zaključi le v primeru, ko testerji zagotovijo, da v programski rešitvi ni več napak in da je programska rešitev primerna za predajo naročniku.

– Predaja programske rešitve naročniku in izvedba beta testiranja

Ko v programski rešitvi testerji ne najdejo več napak in ko razvijalci odpravijo vse napake, ki so se odkrile med testiranjem programske rešitve, lahko podjetje preda programsko rešitev naročniku (Graharji, 2018).

Preden začne naročnik uporabljati programsko rešitev, jo običajno tudi sam testira, tako da

opravi tako imenovano beta testiranje (ang. beta testing). Morebitne napake naročnik sporoči podjetju, ki je zanj razvilo programsko rešitev, da jih odpravi. Cilj beta testiranja je prenos aplikacije k naročniku programske rešitve in da jo preizkusi dejanski uporabnik (Alwan, 2015). Naročnik namreč opravi testiranje na realnih podatkih in morda odkrije še kakšne napake in pomanjkljivosti v delovanju funkcionalnosti v aplikaciji. Povratna informacija o napakah je za razvijalce in testerje koristna, saj bodo lahko nadaljnjim naročnikom predali bolj kakovostne programske rešitve, ki ne bodo več vključevale morebitnih napak. Poleg tega pa pridobijo povratne informacije o tem, katere funkcionalnosti so pomembne za končnega uporabnika in kje morajo biti pazljivi pri razvoju in testiranju programskih rešitev (Izbrano podjetje, 2016a).

Ko razvijalci odpravijo vse napake in pomanjkljivosti v programski rešitvi, lahko naročnik v svojem podjetju namesti aplikacijo v produkcijsko okolje (Izbrano podjetje, 2016a).

– Vzdrževanje in nadgradnja programske rešitve

Na zadnji, šesti stopnji običajno potekajo poprodajne aktivnosti. Podjetje, ki je za naročnika razvilo programsko rešitev, nudi naročniku po predaji programske rešitve pomoč pri uporabi aplikacije tako z vsebinskega kot tudi s tehničnega vidika. Običajno naročnik programske rešitve sklene z izbranim podjetjem vzdrževalno pogodbo, ki je lahko tudi večletna. V njej so zapisani pogoji koriščenja pomoči uporabnikom, cena vzdrževalne ure, število pavšalnega zakupa mesečnih vzdrževalnih ur itd. Zadnja stopnja je pomembna pri gradnji odnosa stranka – podjetje (Alwan, 2015).

V magistrskem delu sem uporabila Porterjevo verigo vrednosti za umestitev temeljnih in podpornih procesov v izbranem podjetju. Stopnje oziroma podprocesi, ki so opisani v prejšnjem poglavju, kjer opisujem življenjski cikel razvoja programskih rešitev, so implementirani v naslednje temeljne procese v izbranem podjetju: v proces razvoja programskih rešitev, v proces testiranja programskih rešitev, v proces pomoči uporabnikom in v proces prodaje.

V tem poglavju je opisno in slikovno prikazana povezava med Porterjevo verigo vrednosti v izbranem podjetju in življenjskim ciklom razvoja programskih rešitev. Za lažje razumevanje bom vsak podproces v življenjskem ciklu razvoja programskih rešitev umestila v določen temeljni proces po Porterjevi verigi vrednosti v izbranem podjetju.

Prva stopnja – zbiranje in analiza zahtev naročnika, druga stopnja – postavitev celotne infrastrukture programske rešitve in tretja stopnja – razvoj programskih rešitev so implementirani v proces razvoja programskih rešitev.

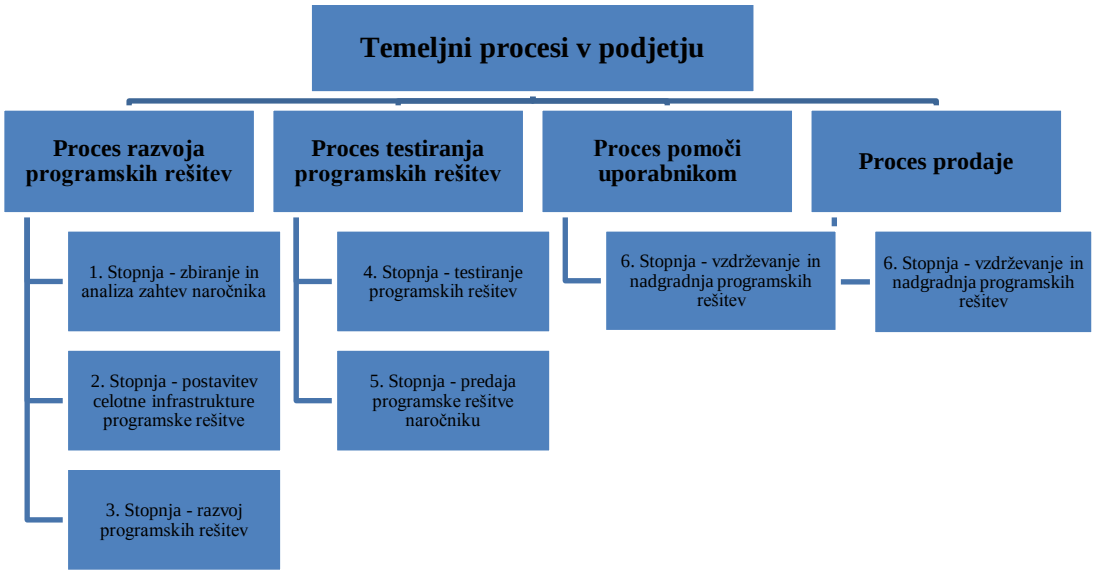
Četrta stopnja – testiranje programskih rešitev in peta stopnja – predaja programske rešitve naročniku sta implementirani v proces testiranja programskih rešitev.

Zadnja, šesta stopnja – vzdrževanje in nadgradnja programskih rešitev – je implementirana

v proces pomoči uporabnikom in v proces prodaje.

Na sliki 22 so prikazani temeljni poslovni procesi po Porterjevi verigi vrednosti v izbranem podjetju in umestitev posamezne stopnje oziroma podprocesa življenjskega cikla razvoja programske rešitve v določen temeljni proces po Porterjevi verigi vrednosti v izbranem podjetju.

Slika 22: Povezava procesov po Porterjevi verigi vrednosti in podprocesi v življenjskem ciklu razvoja programskih rešitev v izbranem podjetju



Vir: lastno delo.

3.3 Metodologija pridobivanja podatkov za potrebo prenove procesa testiranja programskih rešitev

Pred predstavitvijo temeljnih in podpornih procesov v izbranem podjetju in obsežnejšo analizo obstoječega procesa testiranja programskih rešitev je sprva opisana metodologija za pridobivanje podatkov, ki sem jih uporabila pri opredelitvi temeljnih in podpornih procesov v izbranem podjetju, izrisu in analizi procesnega diagrama obstoječega procesa ter pri izdelavi procesnega diagrama prenovljenega procesa.

Temeljne in podporne procese v podjetju sem opredelila in opisala s pomočjo interne dokumentacije, v kateri je bila opredelitev osnovnih procesov, ki se izvajajo v podjetju. Pri opisu posameznega procesa sem vključila tudi informacije, ki sem jih prejela od zaposlenih, vključenih v posamezni proces.

Diagram obstoječega procesa testiranja sem izrisala s pomočjo preučitve interne dokumentacije izbranega podjetja. V podjetju je namreč obstajal opis obstoječega procesa v programu Word in osnovni procesni diagram, izrisan v programu Microsoft Visio. Poleg

opisa obstoječega procesa testiranja sem preučila tudi dokumentacijo o celotnem življenjskem ciklu razvoja programske opreme. Tako sem lažje uvrstila proces razvoja programskih rešitev na zrelostno stopnjo po metodi CMMI in ugotovila, v katere faze življenjskega cikla razvoja programske opreme je implementiran proces testiranja.

Pri opisu problematike in predlogih za izboljšavo obstoječega procesa sem si pomagala z delovnimi izkušnjami kot testerka programskih rešitev, z zapisniki internih sestankov, z analiziranjem uporabniških in tehničnih specifikacij, ki so jih napisali različni testerji in razvijalci ter s pomočjo preučitve procesnega diagrama obstoječega procesa, ki sem ga izrisala v BPMN v programu Bizagi Modeler.

V izbranem podjetju za beleženje časa izvajanja posamezne aktivnosti v celotnem življenjskem ciklu razvoja programskih rešitev za vsako funkcionalnost trenutno uporabljajo aplikacijo Jira. Jiro je razvilo podjetje Atalissian za namen agilnega projektnega vodenja in učinkovitega spremljanja celotnega razvoja programske rešitve. Po analizi podjetja Atlassian aplikacijo Jira podjetja najpogosteje uporabljajo za spremljanje in upravljanje projektov (ATLASSIAN, 2018). Jira je uporabnikom na voljo v treh različnih paketih (ATLASSIAN, 2018):

- Jira Core – namenjena je splošnemu vodenju projektov;
- Jira Software – vključuje osnovno programsko opremo za upravljanje projektov;
- Jira Service desk – namenjena je predvsem za učinkovito upravljanje procesov, ki vključujejo poslovne storitve tistih podjetij, ki se ukvarjajo z razvojem in vzdrževanjem informacijske tehnologije in nudijo uporabnikom pomoč pri uporabi aplikacij.

V izbranem podjetju uporabljajo produkt Jira Software. Za vse funkcionalnosti so določili podprocese, ki potekajo skozi celotni življenjski cikel razvoja funkcionalnosti. Za vsako aktivnost zaposleni, ki so vključeni v proces razvoja in testiranja programske rešitve, zapišejo čas trajanja, ki so ga porabili za izvajanje določene aktivnosti, ter dodajo kratek opis aktivnosti. A je bila večina aktivnosti zgolj časovno opredeljena, opisi aktivnosti pa so bili zelo pomanjkljivi, zato si z njimi nisem mogla pomagati pri načrtovanju procesnega diagrama obstoječega procesa. Podatke o trajanju aktivnosti sem uporabila samo pri časovni analizi obstoječega procesa. V izbranem podjetju aplikacijo Jira uporabljajo izključno za časovno analizo trajanja aktivnosti v življenjskem ciklu razvoja funkcionalnosti, da lahko pri nadaljnjih projektih optimalno razporedijo zaposlene in finančna sredstva.

Proces testiranja v izbranem podjetju so v aplikaciji Jira razdelili na štiri ključne podprocese, ki so predstavljeni v nadaljevanju (Izbrano podjetje, 2016b):

- proces pregleda kode (v Jiri status »Code review«**)** – koda funkcionalnosti je v fazi pregleda;
- proces priprave na alfa testiranje (v Jiri status »Ready for test«) – tester si pripravlja

testno okolje in prebira dokumentacijo za funkcionalnost, ki jo bo testiral;
- proces testiranja funkcionalnosti (izvedba alfa testiranja – v Jiri status »In test«) – tester testira funkcionalnost;
- proces priprave funkcionalnosti za predajo naročniku (v Jiri status »Waiting for a customer«**)** – tester dopolnjuje uporabniška navodila za funkcionalnost, funkcionalnost je pripravljena za predajo naročniku.

Na sliki 23 je prikazan proces testiranja funkcionalnosti, ki so ga opredelili v aplikaciji Jira in je sestavljen iz štirih ključnih podprocesov: proces pregleda kode, proces priprave na alfa testiranje, proces testiranja funkcionalnosti in proces priprave funkcionalnosti za predajo naročniku.

Slika 23: Celotni proces testiranja v izbranem podjetju, opredeljen v aplikaciji Jira

Vir: Izbrano podjetje (2016b).

V celotno analizo obstoječega procesa testiranja sem vključila šestdeset funkcionalnosti.

Analizo virov sem izvedla s pomočjo pregleda šestdesetih funkcionalnosti v aplikaciji Jira. Ugotovila sem, da posamezno funkcionalnost razvija natanko en razvijalec in testira natanko en tester. Za vsako funkcionalnost se izvede tudi pregled kode, ki jo opravi razvijalec, ki ni razvijal programske kode za funkcionalnost. V redkih primerih, npr. da razvijalec ali tester odideta iz podjetja ali pa da sta daljši čas odsotna, prevzame odgovornost za razvoj oziroma testiranje funkcionalnosti nekdo drug zaposlen v podjetju. A tega primera v magistrski nalogi ne bom obravnavala. Torej pri celotnem procesu testiranja sem pri analizi obstoječega procesa uporabila podatek, da so v procesu testiranja za določeno funkcionalnost udeleženi trije zaposleni: razvijalec, ki razvije funkcionalnost in ki naj bi opravil osnovni test funkcionalnosti, tester in razvijalec, ki je zadolžen za pregled kode.

Ker v izbranem podjetju zaradi poslovne skrivnosti o plačah nisem pridobila podatka o plačah zaposlenih, ki so udeleženi v proces testiranja, sem pri analizi virov uporabila povprečne bruto plače za testerje in razvijalce, ki so objavljeni na spletni strani www.placa.si. V izbranem podjetju sem pridobila le podatek o fiksnih mesečnih stroških na zaposlenega, ki v povprečju znašajo okoli 300 EUR. Med fiksne stroške so upoštevani: strošek najema poslovnih prostorov, strošek elektrike, vode, interneta in ogrevanja, strošek pisarniškega materiala, strošek čistilnega servisa, stroški zavarovanja osnovnih sredstev, stroški dejavnosti in ostala zavarovanja.

Analiza virov je podrobneje predstavljena v nadaljevanju magistrskega dela, kjer je opisana celotna analiza obstoječega procesa testiranja.

Podatke za časovno analizo sem črpala iz aplikacije Jira. Kot že omenjeno, se v tej aplikaciji za vsako funkcionalnost nahaja pet ključnih informacij za testerje in razvijalce: skozi katere faze življenjskega cikla je prešla funkcionalnost, v kateri fazi življenjskega cikla se funkcionalnost trenutno nahaja, koliko časa je bilo potrebno za izvedbo določene faze, kdo od razvijalcev je razvijal določeno funkcionalnost in kdo od testerjev jo je testiral.

Za izvedbo časovne analize obstoječega procesa sem izbrala izključno funkcionalnosti, katerih življenjski cikel razvoja se je zaključil in so bile nadgrajene v aplikacije pri naročnikih.

Čas izvajanja posameznih aktivnosti med procesom testiranja sem pridobila tako, da sem si v aplikaciji Jira izbrala šestdeset naključnih funkcionalnosti in jih izvozila v Excel–ovo datoteko. S pomočjo Excel–ovih funkcij sem izračunala povprečni čas izvajanja posameznih aktivnosti in standardne odklone časovnega izvajanja posameznih aktivnosti.

Glede na to, koliko časa se v povprečju izvaja določena aktivnost med procesom testiranja in kdo od udeležencev v procesu jo izvaja, sem naredila tudi stroškovno analizo procesa testiranja.

3.4 Uporaba metode za vrednotenje kakovosti razvoja programskih rešitev v izbranem podjetju

S pomočjo metode za vrednotenje kakovosti razvoja programskih rešitev, ki je opisana v začetku tretjega poglavja, sem določila zrelostno stopnjo razvoja programskih rešitev. To sem storila na podlagi interne dokumentacije, ki sem jo pridobila v izbranem podjetju, s pomočjo pogovorov z zaposlenimi in mojimi delovnimi izkušnjami.

V izbranem podjetju imajo na kratko opisane poteke naslednjih procesov (Izbrano podjetje, 2018a):

- SDLC model, ki je sestavljen iz več stopenj. Vsaka stopnja predstavlja podproces. Celoten proces je opisan na začetku poglavja.
- Proces testiranja programskih rešitev, ki je podrobneje predstavljen v magistrskem delu.
- Proces reševanja zahtevkov s strani naročnikov.
- Proces zaposlovanja.

Vsak proces je predstavljen z osnovnim diagramom poteka.

Po pogovoru z vodilnimi zaposlenimi in po preučitvi interne dokumentacije, izbrano

podjetje po metodi CMMI uvrščam v drugo fazo zrelosti razvoja programskih rešitev. Svojo ugotovitev utemeljim z naslednjimi dejstvi:

- V izbranem podjetju imajo vzpostavljene osnovne procese.
- Imajo delno vzpostavljeno projektno vodenje.
- Procese načrtujejo – zlasti proces razvoja programskih rešitev, ki je ključen proces v podjetju za ustvarjanje dodane vrednosti in je sestavljen iz podprocesov.
- Merijo in spremljajo stroške v procesu razvoja programske rešitve.
- Procesi so vzpostavljeni do te mere, da jih lahko ponovijo na podobnih projektih, kjer želijo doseči primerljive rezultate.
- Končna programska rešitev zadošča zahtevam, ciljem, potrebam in standardom stranke.
- Ni metrik za ocenjevanje uspešnosti opravljenega dela.
- Ni jasno razdelanega procesa pri uvajanju novih sodelavcev, kar posledično vodi v neracionalno izrabo časa zaposlenih.
- Pomanjkanje načrtovanja časovnega okvirja za izdajo programske rešitve naročniku.
- Časovni okvir za razvoj in testiranje programske rešitve ni natančno določen, kar povzroča, da so nekatere programske rešitve in funkcionalnosti v njej zaradi časovne stiske predaje programske rešitve naročniku hitro in pomanjkljivo testirane. Posledično se določene napake pri programski rešitvi pokažejo šele, ko jo uporabljajo končni uporabniki.

3.5 Proces testiranja programskih rešitev

Testiranje programske rešitve v procesu razvoja programske rešitve je podproces, ki se začne že z drugo stopnjo razvoja programske rešitve (ko se vzpostavlja celotna infrastruktura programske rešitve) in se konča s peto stopnjo razvoja programske rešitve (ko se programsko rešitev preda naročniku in ko končni uporabniki izvedejo beta testiranje).

Bistveno vprašanje, ki si ga zastavljajo v podjetjih za razvoj programskih rešitev je, kdaj je najbolj primerno začeti s testiranjem programske rešitve. Če želijo imeti razvojna podjetja kakovostno programsko rešitev s čim manj napakami, potem bi morali proces testiranja začeti že zelo zgodaj, na drugi stopnji življenjskega cikla razvoja programske rešitve. Z zgodaj načrtovanim naborom testnih primerov in jasno razdelanim procesom testiranja, ki poteka skozi različne stopnje razvoja programske rešitve, bi v podjetju znižali stroške razvoja, preprečili odkrivanje večjih napak v funkcionalnostih šele v fazi testiranja programske rešitve ter časovno skrajšali celotni proces razvoja programske rešitve. Proces testiranja programske rešitve je različen od podjetja do podjetja in je odvisen od prvotnega procesa razvoja programske rešitve (Lewis, 2005, str. 26). Npr., če v podjetju uporabljajo klasični življenjski cikel razvoja programske opreme, potem se glavni proces testiranja programske rešitve izvede na četrti stopnji in nadaljuje na peti stopnji, ko naročnik izvede beta testiranje. Če vzamemo za primer procesa razvoja programske rešitve V–model,

potem se testiranje programske rešitve izvaja sočasno s samim razvojem programske rešitve, pri waterfall modelu, ki je precej zastarel model za razvoj programske rešitve, pa se proces testiranja začne šele v fazi testiranja (Ruparelia, 2010). Tako kot imamo na voljo mnogo procesov za razvoj programske opreme, imamo na voljo tudi za proces testiranja programske rešitve na voljo številne procese.

Proces testiranja programskih rešitev v podjetju, ki razvija programske rešitve, sodi med temeljne procese, zato je zelo pomembno, da je dobro definiran in prilagojen glede na celoten proces razvoja programske rešitve. Ker imajo v izbranem podjetju že razvite programske rešitve, ki jih prodajajo na trgu in jih že nekaj let samo nadgrajujejo ter dopolnjujejo z novimi funkcionalnostmi, je pomembno, da stranka v čim krajšem možnem času in po konkurenčni ceni prejme nadgradnjo programske rešitve. Zahteve in pričakovanja stranke so, da dobi nove funkcionalnosti in da je v procesu beta testiranja minimalno število napak. Zato je pomembno, da že v izbranem podjetju dobro izvedejo testiranje funkcionalnosti (Izbrano podjetje, 2016a).

Vsako podjetje, ki razvija programske rešitve, in naročnik programskih rešitev se morata zavedati, da programska rešitev nikdar ne more biti povsem brez napak, ko je enkrat nameščena v produkcijskem okolju naročnika. Pomembno pa je, da je teh napak čim manj in da uporabniku ne onemogočajo uspešno opravljati delovnih nalog prek aplikacije (Izbrano podjetje, 2016a).

V strokovni literaturi najdemo številne definicije testiranja programskih rešitev, nekaj jih navajam v nadaljevanju:

- Testiranje programske opreme je proces preverjanja pravilnosti delovanja aplikacije s ciljem odkriti čim več napak in pomanjkljivosti ter jih odpraviti pred predajo programske rešitve naročniku (The Economic Times, 2018).
- Testiranje programskih rešitev je metoda, ki se uporablja za ocenjevane pravilnosti delovanja funkcionalnosti v aplikaciji, v skladu z zahtevami, zapisanimi v tehnični in uporabniški specifikaciji za delovanje programske rešitve. Izvajalci testiranja programskih rešitev imajo na voljo več vrst testiranja. Odločijo se naj za tisto, ki najbolj ustreza celotnemu življenjskemu procesu razvoja programske rešitve (Izbrano podjetje, 2016a).
- Testiranje programske rešitve je proces, sestavljen iz podprocesov, ki so namenjeni preizkušanju, vrednotenju in ugotavljanju kakovosti programske rešitve. Proces testiranja se mora izvesti v skladu s tehničnimi, poslovnimi, zakonskimi ter uporabniškimi zahtevami (Technopedia, 2018).
- Testiranje programske rešitve je širok proces, sestavljen iz več podprocesov, ki se med seboj povezujejo in dopolnjujejo. Glavni cilj testiranja je določiti kakovost izdelane programske rešitve v skladu s tehnično in uporabniško specifikacijo. Testiranje se izvede prek nabora različnih testnih primerov. Preizkušena programska rešitev mora uspešno prestati vse testne primere v vseh procesih testiranja, da je primerna za predajo

naročniku v produkcijsko okolje (Lewis, 2005, str. 10–11).

- Testiranje programskih rešitev je proces in ne aktivnost. Proces se začne z načrtovanjem testiranja, nadaljuje z načrtovanjem testnih primerov, izvedbo testnih primerov na programski rešitvi, ocenjevanjem kakovosti izhodnih podatkov testnih primerov in zaključi s poročanjem ter dokumentiranjem testnih rezultatov (TRY QA, 2018).

3.6 Obstoječi proces testiranja programskih rešitev

Obstoječi proces testiranja programskih rešitev v izbranem podjetju je opisan glede na pogovore z zaposlenimi, po preučitvi obstoječe interne dokumentacije in glede na moje delovne izkušnje, ko sem bila testerka programskih rešitev. Trenutni proces je predstavljen v obliki AS–IS modela, ki sem ga izrisala v programu Bizagi Modeler.

Pri opisu obstoječega procesa testiranja sem si pomagala s procesom testiranja, ki je implementiran v aplikaciji Jira. Vsak podproces v aplikaciji Jira je podrobneje razdelan in opisan.

Večina testiranja se opravi na četrti stopnji življenjskega cikla razvoja programske rešitve, osnovne teste razvijalci opravijo tudi na tretji stopnji. Ker izbrano podjetje na trgu prodaja aplikacijo, ki vsebinsko in tehnično sodi med zahtevnejše aplikacije, običajno en razvijalec nadgrajuje ali pa razvija eno funkcionalnost. Redko se zgodi, da eno funkcionalnost razvijata dva razvijalca. V primeru, da določen naročnik poda zahtevek za razvoj ali nadgradnjo več manjših enostavnejših funkcionalnosti, običajno kar en razvijalec razvije nabor več enostavnejših funkcionalnosti, kar je časovno približno enako kot bi razvijal eno zahtevnejšo funkcionalnost. Za testiranje ene zahtevnejše ali pa več enostavnejših funkcionalnosti je zadolžen en tester, v redkih primerih dva.

Proces testiranja se odvija v dveh oddelkih, in sicer v razvojnem in tehnološkem oddelku. V procesu so udeleženi trije zaposleni: razvijalec 1 (razvijalec, ki razvije oziroma nadgradi obstoječo funkcionalnost), razvijalec 2 (razvijalec, ki najbolje tehnično in vsebinsko pozna aplikacijo, lahko je tudi vodja projekta) in tester programskih rešitev (v nadaljevanju tester).

Proces se začne, ko razvijalec 1 konča z razvojem nove funkcionalnosti in prek elektronske pošte obvesti razvijalca 2, da je koda funkcionalnosti pripravljena za pregled (ang. code rewiev). Razvijalec 2 po prejetju elektronske pošte pregleda tehnično dokumentacijo funkcionalnosti.

Ko ima dovolj potrebnih informacij o tem, kako naj bi delovala funkcionalnost, izvede validacijo in verifikacijo kode funkcionalnosti. V primeru, ko razvijalec 2 po pregledu kode funkcionalnosti ugotovi, da zadošča vsem standardom, zapisanim v tehnični dokumentaciji in tudi v primeru, da je koda napisana v skladu z internim pravilnikom

razvijanja programske rešitve, potem prek elektronske pošte razvijalca 1 obvesti, da lahko opravi osnovni test funkcionalnosti.

Če razvijalec 2 v kodi najde napake ali pa ugotovi, da koda ni razvita v skladu s tehnično dokumentacijo ali da razvijalec 1 pri razvijanju kode ni upošteval internega pravilnika za razvoj programske rešitve, potem se na tej točki zaključi proces nadaljnjega testiranja kode funkcionalnosti. Koda funkcionalnosti gre nato ponovno v razvoj k razvijalcu 1 (Izbrano podjetje, 2016b). Delček opisanega procesa, ki je prikazan na sliki 25, prikazuje aktivnosti v procesu, prek katerih razvijalec 2 izvede pregled kode. V aplikaciji Jira je delček opisanega procesa implementiran kot podproces pregleda kode – na sliki 24 je podproces označen z zeleno barvo.

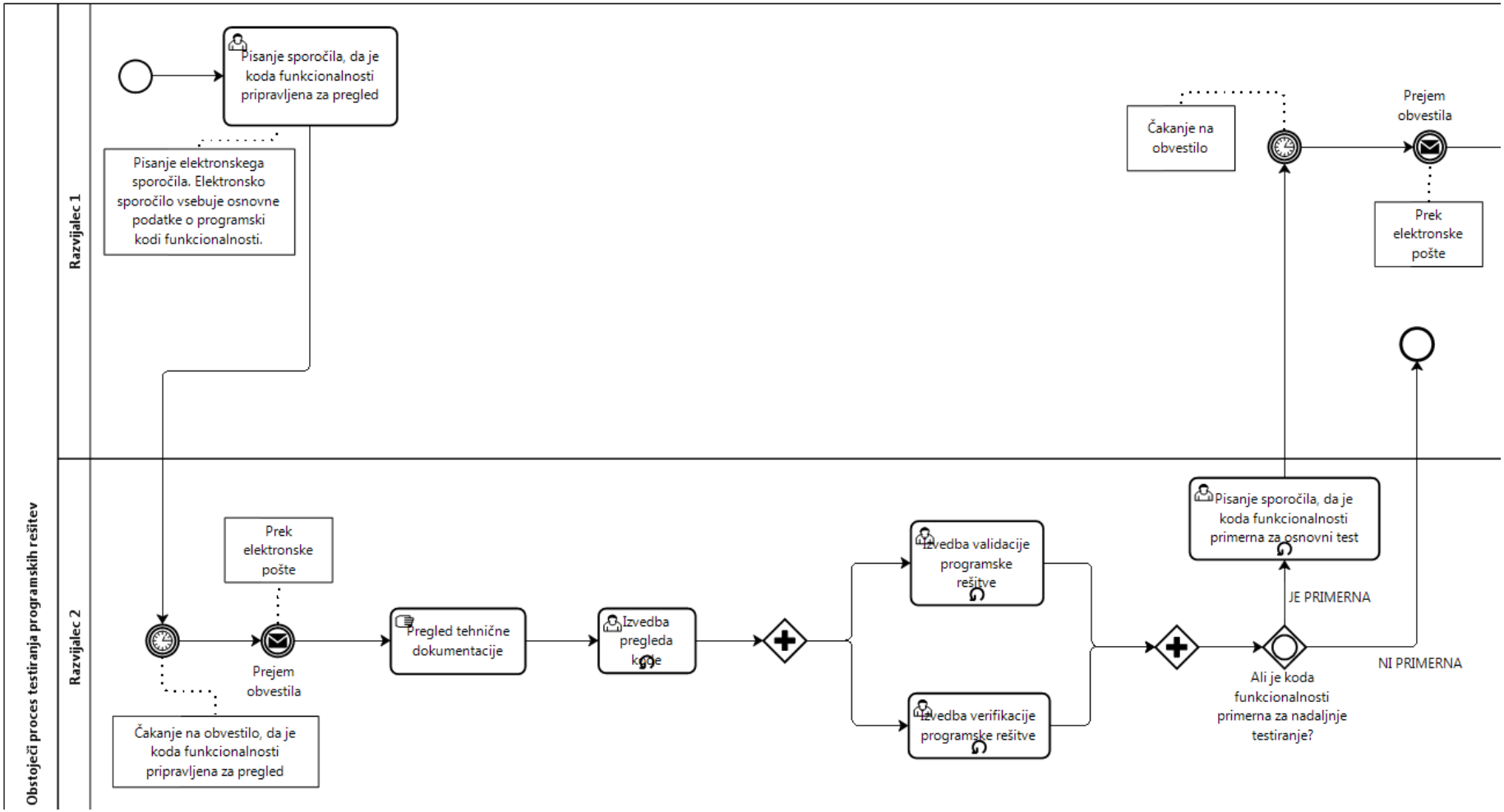
V primeru, ko je koda funkcionalnosti primerna za izvedbo osnovnega testa funkcionalnosti, razvijalec 1 naredi osnovni test funkcionalnosti na vsaj treh do petih različnih testnih primerih v aplikaciji. Če funkcionalnost ne vrne pričakovanih rezultatov za osnovne testne scenarije, se proces testiranja zaključi. Razvijalec 1 mora popraviti funkcionalnost tako, da bo delovala pravilno vsaj za osnovne testne scenarije. Ko funkcionalnost vrne pričakovane rezultate za osnovne testne scenarije, razvijalec 1 pošlje elektronsko pošto testerju, naj začne z izvedbo alfa testiranja.

Delček opisanega procesa, ki je prikazan na sliki 26, prikazuje aktivnosti v procesu, prek katerih razvijalec 1 izvede osnovni test funkcionalnosti. V aplikaciji Jira je delček opisanega procesa implementiran kot podproces pregleda kode – prikazano na sliki 24.

Slika 24: Podproces pregleda kode je označen z zeleno barvo

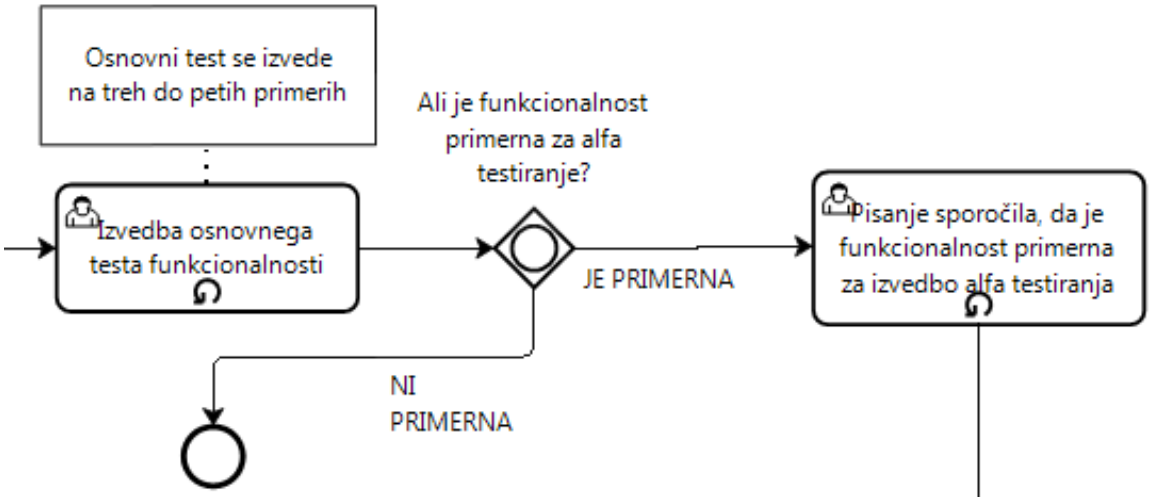
Vir: Izbrano podjetje (2016b).

Slika 25: Obstoječi proces – izvedba pregleda kode



Vir: lastno delo.

Slika 26: Obstoječi proces – izvedba pregleda kode – nadaljevanje



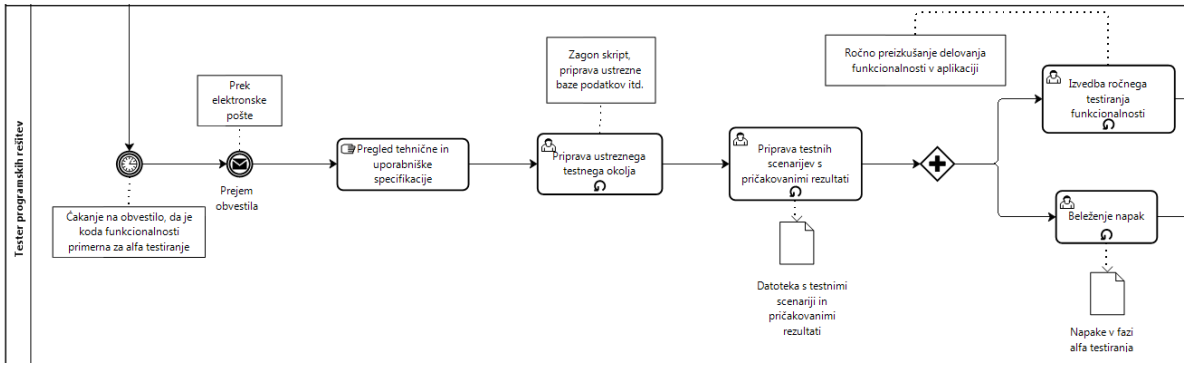
Vir: lastno delo.

Ko tester prejme elektronsko sporočilo od razvijalca 1, da lahko prične z izvedbo alfa testiranja, v prvem koraku preuči tehnično in uporabniško specifikacijo (če ta že obstaja) za funkcionalnost, ki jo bo testiral. V drugem koraku si pripravi ustrezno testno okolje za izvedbo testiranja – pripravi si testno bazo podatkov, nastavi ustrezno verzijo aplikacije itd. Nato v Excelu ali Wordu pripravi testne scenarije s pričakovanimi rezultati. Ko ima pripravljene testne scenarije, začne ročno preizkušati delovanje aplikacije. Med testiranjem si sproti beleži rezultate testnih scenarijev. Delček opisanega procesa, ki je prikazan na sliki 28, prikazuje aktivnosti, ki jih opravi tester, da si pripravi testno okolje ter da opravi alfa testiranje. V aplikaciji Jira je delček opisanega procesa implementiran kot podproces priprave na izvedbo alfa testiranja in podproces testiranja funkcionalnosti – na sliki 27 sta podprocesa označena z zeleno barvo.

Slika 27: Podproces priprave na alfa testiranje in podproces testiranja funkcionalnosti sta označena z zeleno barvo

Vir: lastno delo.

Slika 28: Obstoječi proces – podproces priprave na izvedbo alfa testiranja in podproces testiranja funkcionalnosti



Vir: lastno delo.

V primeru, ko funkcionalnost vrne pričakovane rezultate za vse testne scenarije ali ko tester v delovanju funkcionalnosti ne najde nobene napake, potem dopolni uporabniška navodila in pošlje elektronsko sporočilo razvijalcu 1, da funkcionalnost deluje pravilno za vse testne scenarije. S tem se proces testiranja uspešno zaključi. Funkcionalnost je po uspešno zaključenem alfa testiranju primerna za predajo naročniku. Če tester med alfa testiranjem odkrije napake, po koncu testiranja po elektronski pošti pošlje razvijalcu 1 poročilo o napakah. Proces testiranja se zaključi neuspešno, koda funkcionalnosti zopet pade v fazo razvoja, kjer ima razvijalec 1 nalogo, da odpravi vse napake in pomanjkljivosti, ki jih je tester odkril med alfa testiranjem.

Razvijalec 1 mora v najkrajšem možnem času odpraviti vse napake, ki so se pojavile v prvi iteraciji testiranja. Medtem ko razvijalec 1 odpravlja napake v kodi funkcionalnosti, si tester pripravi še dodatne testne scenarije za testiranje v drugi iteraciji. Po prvi iteraciji testiranja ima namreč boljši vpogled v delovanje funkcionalnosti.

Ko razvijalec 1 popravi kodo, o tem obvesti razvijalca 2, da ponovno opravi pregled kode in celotni proces testiranja se ponovi. Tester drugo iteracijo testiranja prične s testnimi scenariji, ki v prvi iteraciji niso dali pričakovanih rezultatov. V izbranem podjetju je precejšnja težava, da velika večina testerjev ne ponovi testnih scenarijev iz prve iteracije testiranja – tudi tistih ne, za katere je funkcionalnost v prvi iteraciji dala pričakovane rezultate. Ko popravljena funkcionalnost da pričakovane rezultate za testne scenarije iz prve iteracije testiranja, prične tester preizkušati delovanje funkcionalnosti na testnih scenarijih, ki si jih je pripravil za drugo iteracijo. Po zaključku nabora testnih scenarijev tester ponovno prek elektronske pošte obvesti razvijalca 1 o morebitnem obstoju napak. V primeru napak gre funkcionalnost znova v fazo razvoja.

V izbranem podjetju testerji v povprečju izvedejo tri do štiri iteracije testiranja nadgradnje funkcionalnosti, odvisno od njene vsebinske in tehnične zahtevnosti. Za enostavnejše funkcionalnosti v povprečju zadostujejo dve do tri iteracije procesa testiranja, za

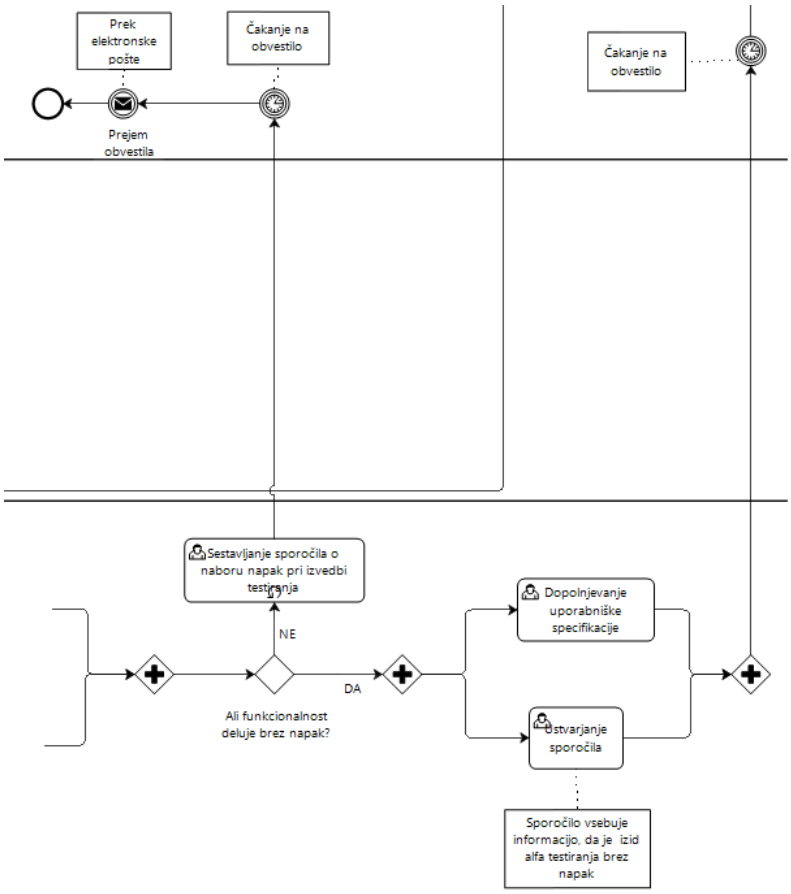
zahtevnejše funkcionalnosti pa je v povprečju potrebnih tri do pet iteracij.

Ko tester pri izvajanju alfa testiranja ne najde več napak, izbrano podjetje naročniku izvede nadgradnjo aplikacije. Del opisanega procesa je prikazan na sliki 30. Delček opisanega procesa je v aplikaciji Jira implementiran kot podproces priprave funkcionalnosti za predajo naročniku – na sliki 29 je podproces označen z zeleno barvo.

Slika 29: Podproces priprave funkcionalnosti za predajo naročniku je označen z zeleno barvo

Vir: lastno delo.

Slika 30: Obstoječi proces – podproces priprave funkcionalnosti za predajo naročniku



Vir: lastno delo.

3.7 Analiza obstoječega procesa

V tem poglavju so podrobneje opredeljene težave in pomanjkljivosti, ki jih ima trenutni proces testiranja programskih rešitev v izbranem podjetju. Pri opisu vsake težave oziroma pomanjkljivosti je navedeno, na kakšen način sem prišla do ugotovitev.

Problematiko obstoječega procesa sem preučila s pomočjo opisanega procesnega diagrama in s pomočjo nekaterih internih gradiv izbranega podjetja, kot so tehnične specifikacije, uporabniška navodila, uporaba aplikacije Jira in pregled Share point–a. Poleg vse pisne dokumentacije pa sem pri opisu problematike črpala informacije tudi prek internih sestankov v izbranem podjetju.

Po pogovoru s testerji in glede na moje delovne izkušnje kot testerka programskih rešitev sem ugotovila, da testerji večinoma sestavljajo testne scenarije kar med testiranjem funkcionalnosti. Sestaviti bi jih namreč morali že v tretji fazi življenjskega cikla razvoja programskih rešitev (tj. v fazi razvoja programske rešitve), in sicer po podrobni preučitvi tehnične in uporabniške specifikacije. Glavni razlog za tovrstno težavo je, da v izbranem podjetju nimajo jasno in podrobneje opredeljenih posameznih faz življenjskega cikla razvoja programskih rešitev. Torej nimajo opredeljeno, katere aktivnosti in delovne naloge se odvijajo v določeni fazi življenjskega cikla razvoja programskih rešitev.

Razvijalci zaradi časovne stiske predaje nadgradnje funkcionalnosti stranki velikokrat ne opravijo osnovnega testiranja, oziroma če ga opravijo, si ne beležijo testnih scenarijev s pričakovanimi rezultati. Če razvijalci ne izvedejo osnovnega testa funkcionalnosti, testerji že po izvedbi nekaj testnih primerov zaključijo s prvo iteracijo testiranja, saj funkcionalnost ne deluje niti za osnovne testne primere. Pri tem testerji izgubijo kar nekaj časa s postavitvijo testnega okolja in dokumentiranjem napak. Omenjena pomanjkljivost je velikokrat izpostavljena na internih sestankih v podjetju, a doslej niso sprejeli nikakršnih resnejših ukrepov, kako jo odpraviti. V aplikaciji Jira pri nobeni od analiziranih funkcionalnosti nisem našla pisnega komentarja, da je funkcionalnost prestala osnovni test programerja, ki je razvijal kodo funkcionalnosti (Izbrano podjetje, 2018a).

Razvijalci naredijo premalo osnovnih testov. Posledično testerji že po nekaj osnovnih testnih primerih lahko zaključijo s prvim krogom testiranja funkcionalnosti, saj funkcionalnost ne vrne pričakovanih rezultatov niti za osnovne testne primere.

Pregleda kode (ang. code review) v večini primerov ne opravi izkušen razvijalec, temveč razvijalec, ki nima vsebinskega in tehničnega znanja o celotni programski rešitvi. Izkušeni razvijalci, ki so hkrati tudi arhitekti drugih programskih rešitev, zaradi časovne stiske velikokrat ne utegnejo pregledati kode za novo oziroma nadgrajeno funkcionalnost, zato prepustijo to delo manj izkušenim razvijalcem. Posledično lahko v fazi alfa testiranja tester odkrije resnejše vsebinske in tehnične napake v programski rešitvi, kar bistveno podaljša čas razvoja in testiranja.

Da v večini primerov pregleda kode ne izvaja izkušen razvijalec, je razvidno iz statusov v programu Jira. Od osemdesetih naključnih pregledov funkcionalnosti je v šestinštiridesetih primerih izvedel pregled kode (v Jiri status »Code review«) izkušen razvijalec. Torej je le v 57,5 odstotka analiziranih funkcionalnosti pregled kode izvedel izkušen razvijalec. V vseh ostalih primerih je pregled kode izvedel manj izkušen razvijalec, ki razvija programsko rešitev, ni pa tudi arhitekt programskih rešitev.

Vse preveč je ročnega testiranja in premalo avtomatskih testov. Posledično so funkcionalnosti, ki vključujejo testiranja velike količine podatkov, testirane pomanjkljivo in s številnimi napakami. Ker v izbranem podjetju ne uporabljajo avtomatskega testiranja, se posledično podaljšuje čas izvedbe procesa testiranja.

Testni scenariji so napisani pomanjkljivo in nerazumljivo. Razvijalci in testerji dokumentirajo testne scenarije s pričakovanimi rezultati na način, da je razumljiv samo njim. Ugotovitev je posledica preučevanja internih dokumentov s testnimi scenariji, ki so jih napisali nekateri testerji in razvijalci.

Glede na to, da določene funkcionalnosti v aplikacijah delujejo podobno, bi lahko uporabili že uporabljene testne scenarije, le da bi vhodne podatke prilagodili vsebini funkcionalnosti. Ker testni scenariji niso dostopni vsem testerjem, se pri vsaki funkcionalnosti sestavijo novi testni scenariji s pričakovanimi rezultati. To predstavlja precejšnjo izgubo časa. S to težavo sem se vsakodnevno srečevala pri testiranju funkcionalnosti, ko sem delala kot testerka programskih rešitev.

Vsak tester ima v svojih mapah na namizju računalnika zabeležene testne scenarije v obliki Wordovih in Excelovih datotek. To je nesmiselno, saj bi bile lahko datoteke objavljene na skupni mreži, kjer bi vsak tester, ki bi testiral podobno funkcionalnost, imel vpogled v testne scenarije drugega testerja in bi tako pospešil proces testiranja. Glede na pogovor s testerji ima vsak tester na svojem lokalnem računalniku dokumentirane testne scenarije s pričakovanimi rezultati.

Testerji ne seznanijo razvijalcev z naborom testnih scenarijev, ki so jih opravili, temveč le z napakami, ki so jih našli med testiranjem. Posledično razvijalci odpravijo napake, ki so jim jih sporočili testerji, ob tem pa lahko pokvarijo kakšno drugo funkcionalnost, ki je pred popravki delovala pravilno. Ker imajo testerji shranjene testne scenarije s pričakovanimi rezultati na svojem lokalnem računalniku, razvijalci nimajo vpogleda v vse testne scenarije, ki jih je opravil tester, zato so pri odpravljanju določene nepravilnosti v funkcionalnosti osredotočeni zgolj na odpravljanje določene napake. Pri tem pa niso pozorni, da nepravilna odprava določene napake lahko povzroči nepravilno delovanje kakega drugega dela funkcionalnosti, ki je pred popravkom deloval pravilno (Izbrano podjetje, 2016a).

Testiranje zahtevnejših funkcionalnosti poteka preveč časa, običajno tester testira funkcionalnost v najmanj treh iteracijah preden se naročniku nadgradi programska rešitev z

novimi oziroma nadgrajenimi funkcionalnostmi. Po preučitvi interne dokumentacije in pregledu nekaterih tehničnih in uporabniških specifikacij sem ugotovila, da je zaradi pomanjkljivo zastavljenega celotnega življenjskega cikla razvoja programskih rešitev dokumentacija pomanjkljiva in tudi nepravilna. To posledično vodi v nerazumevanje delovanja posamezne funkcionalnosti (Izbrano podjetje, 2018a).

Testerji lahko nadaljujejo s testiranjem, ko od razvijalca dobijo obvestilo o odpravi napak, ki so jih odkrili v eni od iteracij testiranja. V večini primerov testerji ne ponovijo testnih scenarijev, ki so jih uporabili v prvi oziroma drugi iteraciji testiranja, temveč le nadaljujejo s testnimi scenariji, ki jih pri testiranju funkcionalnosti še niso uporabili. Posledično lahko končni uporabnik v fazi beta testiranja odkrije napake v delovanju funkcionalnosti, ki so se pojavile med odpravljanjem napak s strani razvijalca. Po pogovoru s testerji in glede na moje delovne izkušnje kot testerka programskih rešitev sem ugotovila, da je časovna stiska glavni razlog, da se testnih scenarijev ne ponovi.

Testerji in razvijalci glede na vsebinsko zahtevnost aplikacije premalo časa namenijo prebiranju tehničnih in uporabniških specifikacij. To razvijalcem otežuje razvoj, osnovno testiranje in razumevanje celotne kode programske rešitve. Testerji si pripravijo pomanjkljive testne scenarije, ki ne vključujejo vseh možnih testnih scenarijev.

Projektni vodja ne postavlja časovnega okvirja za razvoj in testiranje posamezne funkcionalnosti, temveč postavi časovni okvir za nabor funkcionalnosti. Posledično se realni časovni okvir ne ujema s časovnim okvirjem, ki ga določi projektni vodja (v večini primerov se prekorači čas razvoja in testiranja). Podatek o časovnem okvirju za nabor funkcionalnosti sem našla v tehnični dokumentaciji izbranega podjetja. Projektni vodja namreč v tehnični specifikaciji nikoli natančno ne opredeli časovnega okvirja za celoten razvoj in testiranje funkcionalnosti, temveč ga določi le okvirno.

Trenutno v izbranem podjetju nimajo aplikacije, ki bi omogočala beleženje aktivnosti med procesom testiranja in hrambo testnih scenarijev s pričakovanimi rezultati. Aplikacijo Jira uporabljajo zgolj za beleženje časa določene aktivnosti tekom življenjskega cikla razvoja programske rešitve. Ker je proces testiranja temeljni proces v izbranem podjetju, menim, da bi ga bilo smiselno bolje dokumentirati.

Pri prebiranju in analiziranju tehničnih ter uporabniških specifikacij sem prišla do ugotovitve, da nekatere izmed njih niso napisane v skladu z internimi pravili pisanja tehničnih dokumentacij. To močno otežuje razumevanje in prebiranje dokumentacije razvijalcem in testerjem. V izbranem podjetju sem pogrešala predstavitve funkcionalnosti projektnih vodij. Namreč, v drugi fazi življenjskega cikla razvoja programske rešitve (tj. postavitev celotne infrastrukture programske rešitve) bi pripravili ustno predstavitev funkcionalnosti, npr. s pomočjo Power Pointa, kjer bi razvijalcem predstavili predvsem tehnično ozadje funkcionalnosti, testerjem pa vsebinsko plat funkcionalnosti in zahteve naročnika. Menim, da bi se posledično izboljšalo razumevanje same funkcionalnosti, kar bi

pospešilo razvoj in zmanjšanje števila iteracij testiranja. Projektni vodje premalo komunicirajo s testerji in razvijalci programskih rešitev, kar zadeva tehnično in vsebinsko ozadje aplikacije.

V celotnem procesu testiranja ni enotnega pregleda nad testnimi scenariji, ki so jih izvedli razvijalci in testerji. Testerji in razvijalci imajo namreč kar na svojih lokalnih računalnikih shranjene testne scenarije s pričakovanimi rezultati. Po pregledu Share Pointa sem ugotovila, da ga v izbranem podjetju večinoma uporabljajo predvsem za shranjevanje tehnične in uporabniške specifikacije, zgolj izjemoma pa za shranjevanje testih scenarijev s pričakovanimi rezultati.

3.8 Predlog prenove procesa testiranja programskih rešitev

Glede na to, da je proces testiranja eden temeljnih procesov pri nadgradnji programskih rešitev, bom v primeru izbranega podjetja natančno opredelila vloge zaposlenih v procesu testiranja, vsako fazo testiranja in vrsto testiranja med celotnim procesom testiranja. V vsaki fazi testiranja se uporablja določena vrsta testiranja.

Na spletu obstaja veliko že izdelanih procesov testiranja programskih rešitev, ki jih lahko podjetja implementirajo v svoje podjetje. Pomembno je, da podjetja izberejo tak proces testiranja programskih rešitev in ustrezno vrsto testiranj, da kar najbolj sovpadajo z njihovim procesom razvoja programske rešitve.

Preden predstavim TO–BE model v BPMN v programu Bizagi Modeler, bom predstavila že obstoječi proces testiranja programskih rešitev, ki sem ga našla na spletu. Je najbolj primeren za izbrano podjetje glede na vsebinsko in tehnično zahtevnost programske rešitve. Obstoječi proces sem uporabila pri izgradnji TO–BE modela v programu Bizagi Modeler in izdelavi aplikacije v Bizagi Studio.

Proces testiranja programskih rešitev, ki bo osnova za izgradnjo prenovljenega procesa, je sestavljen iz naslednjih faz (HN Computing, 2018):

- testiranje komponent oz. enot (ang. unit testing),
- integracijsko testiranje (ang. integration testing),
- sistemsko testiranje (ang. system testing) in
- testiranje sprejemljivosti (ang. acceptance testing), ki je razdeljeno na dve fazi, in sicer na alfa in beta testiranje.

Proces testiranja, ki poteka v štirih zaporednih fazah je prikazan tudi na sliki 31.

Slika 31: Proces testiranja programskih rešitev v štirih fazah

Vir: Povzeto in prirejeno po TRY QA, (2018).

Testiranje komponent oz. enot (ang. unit testing): To fazo testiranja opravi razvijalec, ki je razvijal programsko rešitev in najbolj razume delovanje kode, ki jo je napisal. Na tej stopnji razvijalec napiše test testiranja komponent, v katerega vključi vse možne testne scenarije, ki zajamejo testiranje vseh razredov in metod znotraj razredov. Pisanje tovrstnega testa je hitro in enostavno, poleg tega pa omogoča ločeno testiranje razredov in tudi ločeno testiranje metod znotraj razredov (SMART BEAR, 2018). Glavni cilj tega testiranja je odkrivanje napak v delovanju posameznih razredov in metod. S testnimi scenariji razvijalec preveri, ali npr. določena metoda znotraj razreda vrne vnaprej zapisane pričakovane rezultate (Lewis, 2005, str. 39).

Za lažje razumevanje faze testiranja komponent je v nadaljevanju naveden primer v programskem jeziku C#. Nek razvijalec je v programski kodi ustvaril razred uporabnik (ang. class User), katerega zapis je prikazan na sliki 32.

Znotraj razreda je napisal metodo, ki vrne ime in priimek uporabnika (ang. def full_name), katere zapis je ravno tako prikazan na sliki 32.

Slika 32: Prikaz razreda class User in metode full_name

```
# app/models/user.rb
class User < ActiveRecord::Base
  attr_accessible :first_name, :last_name


  def full_name
    [first_name, last_name].reject(&:blank?).join(' ')
  end
end
```

Vir: Meczekalska (2015).

Ker je želel preveriti, ali napisana metoda vrne željene rezultate, je uporabil testiranje komponent (Meczekalska, 2015). Testna scenarija in pričakovana rezultata sta v rdečem okvirju na sliki 33.

Slika 33: Testna scenarija in pričakovana rezultata za metodo full_name

```ruby
# spec/models/user_spec.rb
require 'spec_helper'

describe User do
  describe '#full_name' do
    let(:normal_user) { build_stubbed :user, first_name: 'John', last_name: 'Doe' }
    let(:nameless_user) { build_stubbed :user, first_name: ' ', last_name: ' ' }

    it 'returns user name' do
      expect(normal_user.full_name).to eq 'John Doe'
    end

    it 'skips blank values' do
      expect(nameless_user.full_name).to eq ''
    end
  end
end
```

Vir: Meczekalska (2015).

Integracijsko testiranje (ang. integration testing): Po uspešno izvedenem testiranju komponent je naloga razvijalca, da opravi še integracijsko testiranje. Za razliko od testiranja komponent, kjer se testirajo enote in moduli (oziroma razredi in metode) ločeno drug od drugega, se to testiranje izvede na celotni skupini združenih enot in modulov. Namen tega testiranja je, da se odkrijejo napake in pomanjkljivosti v interakciji med posameznimi enotami oziroma moduli in da se preveri, če sistem kot celota deluje pravilno (Lewis, 2005, str. 84).

Za lažje razumevanje integracijskega testiranja je v nadaljevanju primer, ki sem ga navedla pri testiranju komponent. Primer prikazuje, kako se izvaja testiranje posamezne metode znotraj razredov, torej če metoda kot samostojna komponenta vrne pričakovan rezultat glede na napisani testni scenarij.

Pri integracijskem testiranju npr. razvijalec preverja, ali se ime in priimek na novo registriranega uporabnika spletne trgovine pravilno shrani. Razvijalec mora preveriti, če metoda *full_name* deluje pravilno ob registraciji novega uporabnika v sistem in če je pravilno povezana z grafičnim vmesnikom aplikacije (Meczekalska, 2015). Na sliki 34 je prikazan primer zgoraj opisanega integracijskega testiranja.

Slika 34: Primer integracijskega testiranja

```
require 'spec_helper'

feature 'User profile page' do
  let!(:user) { create :user, first_name: 'John', last_name: 'Doe' }
  scenario 'displays user fullname' do
    visit users_path, id: user.id
    expect(page).to have_content user.full_name
  end
end
```

Vir: Meczekalska (2015).

Sistemsko testiranje (ang. system testing): V tej fazi testiranja razvijalec preizkuša delovanje celotne programske rešitve v testnem okolju, ki naj bi bilo čim bolj podobno produkcijskemu okolju naročnika (Lewis, 2005, str. 56).

Testni scenariji morajo biti zato sestavljeni bolj premišljeno in so hkrati tudi bolj kompleksni. Testni primeri preverjajo delovanje procesov v programski rešitvi, če programska rešitev kot celota ustreza zahtevam, zapisanim v tehnični specifikaciji itd. (HN Computing, 2018).

Testiranje sprejemljivosti (ang. acceptance testing): To fazo testiranja običajno opravijo tako testerji v podjetju, kjer so razvijali programsko rešitev, kot tudi naročniki programskih rešitev. Testiranje sprejemljivosti je sestavljen iz dveh vrst testiranj: alfa (ang. alpha testing) in beta testiranja (ang. beta testing) (Lewis, 2005, str. 76).

Alfa testiranje: Ta vrsta testiranja poteka v podjetju, ki je razvilo programsko rešitev. Izvede jo neodvisna testna skupina testerjev, ki dobro pozna vsebino aplikacije in specifikacijo naročnikovih zahtev. Cilj alfa testiranja je, da se testerji postavijo v vlogo končnega uporabnika in pri testiranju aplikacije uporabijo testne scenarije, ki so čim bolj podobni realnemu poslovnemu svetu naročnika (Lewis, 2005, str. 36).

Beta testiranje: Podjetje, ki je razvilo programsko rešitev, po uspešno zaključenem alfa testiranju preda programsko rešitev naročniku. Nato končni uporabniki testirajo aplikacijo na realnih podatkih in primerih iz njihovega poslovnega okolja, običajno na testnem okolju. Po končanem testiranju končni uporabniki pošljejo nabor morebitnih napak in pomanjkljivosti podjetju, ki za njih razvija programsko rešitev, da napake odpravi (Lewis, 2005, str. 32). V primeru, da napak ni, začnejo končni uporabniki aplikacijo uporabljati v produkcijskem okolju. V nasprotnem primeru podjetje, ki je razvilo programsko rešitev oziroma izvedlo nadgradnjo, odpravi napake in ponovi celotni proces testiranja, od testiranja komponent do beta testiranja (Lewis, 2005, str. 32).

Opis beta testiranja je podan zaradi boljšega razumevanja celotnega procesa testiranja. Pri izrisu TO–BE modela ga ne bom upoštevala, saj ni neposredno vključen v proces testiranja znotraj izbranega podjetja. Izvede ga naročnik programskih rešitev.

Vrste testiranj, ki so uporabljene pri posamezni fazi testiranja programske rešitve, so podrobneje predstavljene v nadaljevanju.

Strukturno testiranje ali metoda bele škatle (ang. white–box testing): To vrsto testiranja izvajajo razvijalci. Z njo preverijo, ali napisana koda za programsko rešitev deluje pravilno oziroma natančneje povedano, ali rekurzije, zanke, metode itd. delujejo v skladu s pričakovanimi rezultati. Razvijalec si sestavi testne scenarije s pričakovanimi rezultati (Lewis, 2005, str. 30).

Če želi razvijalec preveriti, kako npr. deluje določena zanka v metodi ali funkciji, jo mora preizkusiti na različnih testnih scenarijih: npr. da se zanka v metodi ne izvede nikoli, da se ponovi enkrat, dvakrat ali večkrat, pri tem pa spremlja izhodne rezultate delovanja metode (Lewis, 2005, str. 32).

V izbranem podjetju naj strukturno testiranje uporabijo v fazi testiranja komponent in v fazi integracijskega testiranja.

Regresijsko testiranje (ang. regression testing): To vrsto testiranja je potrebno opraviti vsakokrat, ko razvijalec spreminja, dopolnjuje in odpravlja napake v programski kodi. Zelo pomembno je, da se jo uporabi, ko se koda pogosto spreminja: npr., ko se dodaja nove funkcionalnosti, ko se odpravlja rizične napake v delovanju določene funkcionalnosti, ki bi lahko vplivale na več ostalih funkcionalnosti itd. Po odpravi napak je potrebno delovanje programske rešitve ponovno testirati na testnih scenarijih, ki so že bili uporabljeni za preverjanje pravilnosti delovanja programske rešitve in so nekoč že dali pričakovane rezultate. Regresijsko testiranje je lahko zelo zahtevno in zamudno, saj mora razvijalec preveriti, kako so določene spremembe (npr. v določeni funkcionalnosti) vplivale na ostale funkcionalnosti. Pri tem mora ponoviti testne scenarije in sestaviti nove testne scenarije, pri katerih bo upošteval spremembe v kodi. Obseg regresijskega testiranja je odvisen od količine spremenjene kode, pa tudi od spremenjenih, nadgrajenih, popravljenih funkcionalnosti itd. (Lewis, 2005, str. 134).

V izbranem podjetju naj regresijsko testiranje uporabijo v fazi testiranja komponent in v fazi integracijskega testiranja.

Funkcionalno testiranje (ang. functional testing): Pri tej vrsti testiranja se preverja, ali metode v programski rešitvi delujejo kot je zapisano v funkcionalni specifikaciji in ostalih dokumentih, ki so nastali pri načrtovanju programske rešitve oziroma njeni nadgradnji. Pri funkcionalnem testiranju se uporablja metoda črne škatle (ang. black–box method), ki obravnava kodo programa kot »črno škatlo«. To pomeni, da testerja aplikacije zanima zgolj delovanje funkcionalnosti v aplikaciji in ne na kakšen način je napisana koda

funkcionalnosti. Naloga testerja je, da preveri, če v aplikaciji pravilno delujejo funkcionalnosti, oziroma če testni scenariji vrnejo pričakovane rezultate (Lewis, 2005, str. 452).

Vzemimo praktični primer delovanja registracije novega uporabnika v aplikacijo: Tester preizkuša delovanje funkcionalnosti tako, da razmišlja o uporabi funkcionalnosti z vidika uporabnika. Nabor testnih scenarijev je sestavljen na način, da vrnejo pričakovane rezultate. Primer testiranja funkcionalnosti opravimo na primeru ustvarjanja gesla za novega uporabnika. Vsak na novo registrirani uporabnik si mora za vstop v aplikacijo ustvariti geslo, ki naj vsebuje npr. najmanj pet znakov. Pri tem mora upoštevati, da uporabi kombinacijo velikih in malih črk ter številk. Primer testnega scenarija: če uporabnik vpiše manj kot pet znakov, mu sistem javi napako, da mora geslo za vstop v spletno trgovino vsebovati najmanj pet znakov, ki so kombinacija velikih in malih črk ter številk. Ko tester preizkuša funkcionalnost za testni scenarij za primer gesla z manj kot petimi znaki, se mu mora izpisati obvestilo o napaki. V primeru, da se geslo kljub vsemu ustvari, metoda za ustvarjanje gesla ne deluje pravilno in ne izpolnjuje zahtev, ki so zapisane v tehnični dokumentaciji aplikacije, zato je potrebno metodo popraviti (Guru99, 2018).

V izbranem podjetju naj funkcionalno testiranje uporabijo v fazi testiranja sprejemljivosti, natančneje pri alfa testiranju.

Nefunkcionalno testiranje (ang. non–functional testing): Za razliko od funkcionalnega testiranja, ki preverja pravilnost delovanja aplikacije oziroma njene funkcionalnosti, se pri nefunkcionalnem testiranju testira zanesljivost, uporabnost, zmogljivost itd. programske rešitve (Guru99, 2018).

V strokovni literaturi najdemo veliko vrst nefunkcionalnega testiranja. V nadaljevanju so opisane tiste, ki so najprimernejše za testiranje programskih rešitev v izbranem podjetju.

- Testiranje uporabnosti (ang. usability testing): Pri tej vrsti testiranja testerji preizkušajo, če je aplikacija uporabniku prijazna in skušajo odgovoriti na naslednja ključna vprašanja (Lewis, 2005, str. 207):
    - Ali uporabnik zna uporabljati osnovne funkcionalnosti v aplikaciji, ko jih prvič uporablja?
    - Kako hitro se uporabnik nauči uporabljati funkcionalnosti v aplikaciji?
    - Ali uporabnik rad uporablja aplikacijo?
    - Ali je grafični vmesnik uporabniku prijazen in si zapomni nastavljeno obliko ter delovanje funkcionalnosti, tudi ko aplikacija ni v uporabi?
    - Kakšne napake lahko naredijo uporabniki pri uporabi aplikacije?
- Testiranje prenosljivosti (ang. portability testing): Pri tej vrsti testiranja se testira prenosljivost aplikacije iz enega programskega okolja v drugega. Npr. kako bo vplivala prenosljivost aplikacije iz operacijskega sistema Windows XP v Windows 10 (Guru99,

2018).

- Varnostno testiranje (ang. security testing): Ker v izbranem podjetju razvijajo programske rešitve, v katerih naročniki shranjujejo pomembne podatke, je ključno, da se varnostno testiranje izvede večkrat in ob večjih nadgradnjah ali pa spremembah v programski rešitvi. Pomembno je predvsem, ali trenutni informacijski sistem varuje vse pomembne podatke in funkcionalnosti v aplikaciji. Primer varnostnega testiranja je, ali je dovoljena uporaba aplikacije brez uporabniškega imena in gesla (Lewis, 2005, str. 206).
- Testiranje obsega (ang. volume testing): To testiranje se nanaša na testiranje programske rešitve z določeno količino podatkov (npr. preverja se velikost baze podatkov, velikost določene datoteke, v katero se shranijo podatki pri izvozu iz aplikacije itd.). Testiranje poteka tako, da se velikost baze podatkov poveča, nato se naredi npr. več izvozov podatkov iz aplikacije in se preveri, ali so se podatki shranili v bazo po povečanju prostora v bazi (Lewis, 2005, str. 204). Ta oblika testiranja je v izbranem podjetju zelo pogosta, saj naročniki na dnevni ravni izvažajo in shranjujejo velike količine podatkov iz aplikacije v bazo podatkov.
- Testiranje obnovitve (ang. recovery testing): S to vrsto testiranja preverimo, kako hitro se delovanje aplikacije vrne v prvotno stanje, npr. po okvari programske ali strojne opreme, po izpadu električnega toka itd. (Lewis, 2005, str. 209).
- Testiranje vzdržljivosti (ang. endurance testing): To testiranje je potrebno izvesti za prikaz delovanja celotne aplikacije v daljšem časovnem obdobju. Če aplikacijo testiramo eno uro, lahko deluje v skladu z našimi pričakovanji, če pa bi jo testirali deset ur, bi se lahko pojavile različne težave (npr. puščanje pomnilnika, kar je posledica napačno sprogramiranih aplikacij itd.) (Guru99, 2018).
- Testiranje obremenitve (ang. load testing): To vrsto testiranja se uporablja pogosto, saj se preizkuša delovanje aplikacije pod pričakovanimi in tudi nepričakovanimi obremenitvami. Primer takšnega testiranja je, da preverimo, kaj se zgodi, če v spletni trgovini hkrati odda naročilo tisoč strank ali pa, če odda naročilo sto tisoč strank (npr. kakšen je odzivni čas omrežja, ali se v sistem zabeležijo vsa naročila itd.) (Lewis, 2005, str. 37).

V izbranem podjetju naj nefunkcionalno testiranje uporabijo v fazi testiranja sprejemljivosti pri izvedbi alfa testiranja in v fazi sistemskega testiranja. Natančneje, naj pri alfa testiranju uporabijo naslednje vrste nefunkcionalnega testiranja: testiranje uporabnosti, testiranje obsega in testiranje obremenitve.

Pri sistemskem testiranju naj izvedejo naslednje vrste nefunkcionalnega testiranja: testiranje prenosljivosti, varnostno testiranje, testiranje obnovitve in testiranje vzdržljivosti.

V tabeli 1 je prikazano, v kateri fazi testiranja se uporabi določena vrsta testiranja in kdo od zaposlenih je zadolžen za izvedbo testiranja v posamezni fazi.

Tabela 1: Faze in vrste testiranj, ki so uporabljene v posamezni fazi testiranja

Faza testiranja	Vrsta testiranja	Izvajalec testiranja
Testiranje komponent	metoda bele škatle, regresijsko testiranje	razvijalec
Integracijsko testiranje	metoda bele škatle, regresijsko testiranje	razvijalec
Sistemsko testiranje	nefunkcionalno testiranje (testiranje prenosljivosti, varnostno testiranje, testiranje obnovitve, testiranje vzdržljivosti)	razvijalec
Testiranje sprejemljivosti (alfa testiranje)	funkcionalno testiranje (uporaba metode črne škatle) ali nefunkcionalno testiranje (testiranje uporabnosti, testiranje obsega in testiranje obremenitve)	tester programskih rešitev
Testiranje sprejemljivosti (beta testiranje)	nefunkcionalno testiranje (ker testiranje izvede končni uporabnik, v izbranem podjetju nimajo neposrednega vpogleda v vrste testiranja)	končni uporabnik programske rešitve

Vir: lastno delo.

Preden predstavim prenovljeni proces testiranja programskih rešitev v izbranem podjetju v obliki TO–BE modela, naj podam še konkretne predloge, s katerimi bi v izbranem podjetju izboljšali celotni življenjski cikel razvoja programske rešitve in ne le procesa testiranja programske rešitve.

Ključni ukrepi, ki so opisani v nadaljevanju, izhajajo iz problematike obstoječega procesa testiranja, uporabe novih faz in vrst testiranja, ki so opisane na začetku poglavja, po analiziranju uporabe aplikacije Jira ter po preučitvi interne dokumentacije v izbranem podjetju, v kateri je bilo navedenih že nekaj predlogov za izboljšanje procesa testiranja.

Predloge za izboljšanje procesa sem oblikovala po pogovoru s projektnimi vodji, arhitekti programskih rešitev in vodjo tehnološkega oddelka, ki ima pregled nad celotnim procesom testiranja v izbranem podjetju. Pomagala sem si tudi z interno dokumentacijo, natančneje z zapisniki sestankov.

Predlogi za izboljšanje procesa testiranja so naslednji:

- Vodja projekta naj pred fazama razvoja in testiranja funkcionalnosti opredeli časovni okvir za razvoj ter testiranje vsake funkcionalnosti posebej. Projektni vodja mora v drugi fazi življenjskega cikla razvoja programskih rešitev (tj. v fazi postavitve celotne infrastrukture) v tehnični specifikaciji nujno natančno določiti število ur za celotni razvoj in testiranje določene funkcionalnosti oziroma nabor funkcionalnosti. Kot sem

že omenila, pri obstoječem procesu razvoj in testiranje velike večine funkcionalnosti preseže časovni okvir, ki je določen v tehnični specifikaciji.

- Vodja projekta naj pred izvajanjem celotnega življenjskega cikla programske rešitve določi, kateri razvijalci in testerji programskih rešitev bodo sodelovali na določenem projektu. Ker bi točno določena skupina razvijalcev in testerjev sodelovala na projektu v celotnem življenjskem ciklu nastajanja programske rešitve, bi se njena kakovost izboljšala, saj bi bili vsi sodelujoči bolje seznanjeni s celotnim življenjskim ciklom programske rešitve.
- V obstoječem procesu je to velika težava, saj projektni vodja pri postavitvi celotne infrastrukture funkcionalnosti določi le enega testerja, ne glede na obsežnost in zahtevnost razvoja funkcionalnosti. To predstavlja veliko težavo, saj je zaradi časovne stiske pri predaji naročniku programska rešitev večkrat (pre)hitro in pomanjkljivo testirana. Projektni vodja naj v sodelovanju z arhitektom programskih rešitev določi težavnost in obseg določne funkcionalnosti oziroma nabora funkcionalnosti. Temu naj prilagodi število testerjev, ki bodo spremljali celotni življenjski cikel razvoja in ki bodo vključeni v proces testiranja funkcionalnosti. Posledično se bo izboljšalo tako vsebinsko kot tehnično poznavanje funkcionalnosti, boljši pa bo tudi izid testiranja funkcionalnosti.
- Pregled kode mora opraviti izključno izkušen razvijalec, saj bi s tem preprečili odkrivanje večjih vsebinskih in tehničnih napak v kasnejših fazah procesa testiranja programskih rešitev. S tem bi skrajšali čas in stroške procesa testiranja ter naročniku zagotovili kakovostnejšo nadgradnjo programske rešitve. Težava je opisana v obstoječem procesu testiranja programskih rešitev izbranega podjetja.
- Ko izkušeni razvijalec potrdi ustreznost in pravilnost kode, mora eden od razvijalcev izbrati faze ter vrste testiranja, ki jih bo uporabil v vsaki fazi.
- Trenutno razvijalci, ki so razvili določeno funkcionalnost, v obstoječem procesu ne shranijo testnih scenarijev s pričakovanimi rezultati na skupni mreži ali pa na SharePoint–u. V prenovljenem procesu sem natančno opredelila, katere faze in vrste testiranja bi moral opraviti razvijalec. S tem, ko bi opravil pomembnejši del testiranja, bi se skrajšal proces testiranja funkcionalnosti. Posledično tudi testerji ne bi izgubljali časa s prijavljanjem osnovnih napak, ki se ne bi zgodile, če bi razvijalec opravil testiranje komponent in integracijsko testiranje (ki sta obvezni fazi testiranja). V obstoječem procesu zagotovo manjkata ti dve fazi testiranja.
- Trenutno testerji za funkcionalnosti (tako enostavne kot zahtevnejše) izvedejo v povprečju tri do štiri iteracije alfa testiranja (kar izhaja iz aplikacije Jira). To povečuje stroške dela zaposlenih in zmanjšuje dobiček podjetja. Izbranemu podjetju predlagam, naj testerji izvedejo največ dve iteraciji testiranja. Razvijalci bi namreč že opravili velik del testiranja v zgoraj navedenih fazah testiranja programske rešitve, pa tudi testni scenariji s pričakovanimi rezultati bi bili na voljo v aplikaciji Bizagi Studio. Tako bi si tester lahko pripravil nabor tistih testnih scenarijev, ki so ključni z uporabniškega vidika in bi pregled kode opravljal izključno izkušen razvijalec.
- Testerji pri testiranju programskih rešitev nikdar ne morejo zagotoviti, da so našli vse

napake. Zato je povsem dovolj, da se izvedeta največ dve iteraciji testiranja s premišljeno načrtovanimi testnimi scenariji, tako s strani razvijalcev kot s strani testerjev. V splošnem velja, da izid testiranja programske rešitve ni odvisen od števila iteracij procesa testiranja, temveč od kakovostnih testnih scenarijev s pričakovanimi rezultati. Ne glede na to, ali v izbranem podjetju za obsežnejšo in zahtevnejšo funkcionalnost izvedejo pet ali celo šest iteracij testiranja, v nobenem primeru ne morejo zagotoviti, da funkcionalnost deluje popolnoma brez napak, tako tehničnih kot vsebinskih.

- Ker je proces testiranja v izbranem podjetju kompleksen in zahteven, predlagam, da v ta namen uporabljajo aplikacijo, v katero bo implementiran celotni proces testiranja programske rešitve.
- Trenutno v izbranem podjetju uporabljajo aplikacijo Jira, ki je namenjena hitremu pregledu celotnega življenjskega cikla določene funkcionalnosti. Ker je že sam proces testiranja zelo kompleksen, je smiselno, da sami razvijejo oziroma uporabijo aplikacijo, v katero bo implementiran proces testiranja programskih rešitev.

Navedeni predlogi za izboljšanje procesa testiranja so primerni, ker sem jih oblikovala s pomočjo zaposlenih, ki zelo dobro poznajo celoten proces razvoja programskih rešitev in imajo bogate delovne izkušnje pri izboljšanju podprocesov v celotnem procesu razvoja programske rešitve. Glede na to, da se izbrano podjetje sooča s časovno stisko pri prenovi in digitalizaciji procesov, menim, da bi predlogi, ki so bili oblikovani na podlagi temeljite preučitve obstoječega procesa testiranja, trenutno najbolj primerni za prenovljeni proces testiranja.

Glede na opis in problematiko obstoječega procesa sem izdelala prenovljeni proces, in sicer v opisni obliki ter v obliki TO–BE modela v programu Bizagi Modeler.

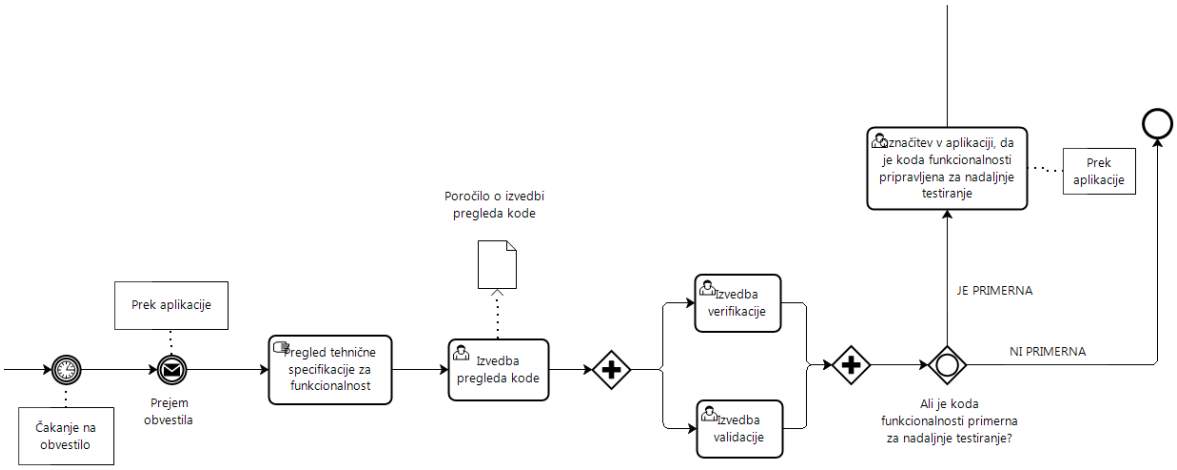
Prenovljeni proces se – tako kot obstoječi proces – odvija v dveh oddelkih, in sicer v razvojnem in tehnološkem oddelku. V procesu so udeleženi trije zaposleni: razvijalec 1 (tj. razvijalec, ki razvije oziroma nadgradi obstoječo funkcionalnost), razvijalec 2 (tj. razvijalec, ki najbolje pozna tehnično in vsebinsko plat aplikacije, lahko je tudi vodja projekta) in tester programskih rešitev (v nadaljevanju tester).

Proces se začne, ko razvijalec 1 konča z razvojem nove funkcionalnosti in v aplikaciji, v kateri je integriran proces testiranja programskih rešitev, ustvari novo nalogo (ang. task) za razvijalca 2. Aplikacija je povezana z elektronsko pošto, tako da je razvijalec 2 prek elektronske pošte avtomatično obveščen o tem, da je koda funkcionalnosti pripravljena za pregled. Pregled kode se pri prenovljenem procesu bistveno ne razlikuje od obstoječega procesa, saj je dobro zasnovan že v obstoječem procesu. Pri prenovljenem procesu mora razvijalec 2 podrobneje dokumentirati pregled kode v primeru, da ni ustrezna za nadaljnje testiranje. V izbranem podjetju bi moral pregled kode nujno izvajati izključno izkušen programer oziroma projektni vodja, ki tako tehnično kot vsebinsko najbolje pozna aplikacijo.

Razvijalec 2 po prejetem obvestilu (ki ga lahko vidi kot elektronsko pošto ali pa v aplikaciji) pregleda tehnično dokumentacijo funkcionalnosti. Ko ima dovolj potrebnih informacij o tem, kako naj bi funkcionalnost delovala, izvede validacijo in verifikacijo kode funkcionalnosti. Če razvijalec 2 po pregledu kode ugotovi, da zadošča vsem standardom, zapisanim v tehnični dokumentaciji funkcionalnosti, in če je koda napisana v skladu z internim pravilnikom razvijanja programske rešitve, potem v aplikaciji označi, da je koda funkcionalnosti pripravljena za nadaljnje testiranje. Razvijalec 1 o tem dobi obvestilo prek elektronske pošte ali pa ga vidi v aplikaciji.

Če razvijalec 2 v kodi ne najde nobene napake ali pa ugotovi, da koda ni razvita v skladu s tehnično dokumentacijo oziroma da razvijalec 1 pri razvijanju kode ni upošteval internega pravilnika za razvoj programske rešitve, potem se na tej točki zaključi proces nadaljnjega testiranja kode funkcionalnost. Koda funkcionalnosti ponovno preide v fazo razvoja. V tem primeru razvijalec 2 v aplikaciji označi, da koda funkcionalnosti ni primerna za nadaljnje testiranje, obenem pa priloži datoteko z napakami in pomanjkljivostmi kode, ki jih mora razvijalec 1 odpraviti. Na sliki 35 je prikazan del predloga prenovljenega procesa z aktivnostmi, ki so potrebne, da se izvede pregled kode.

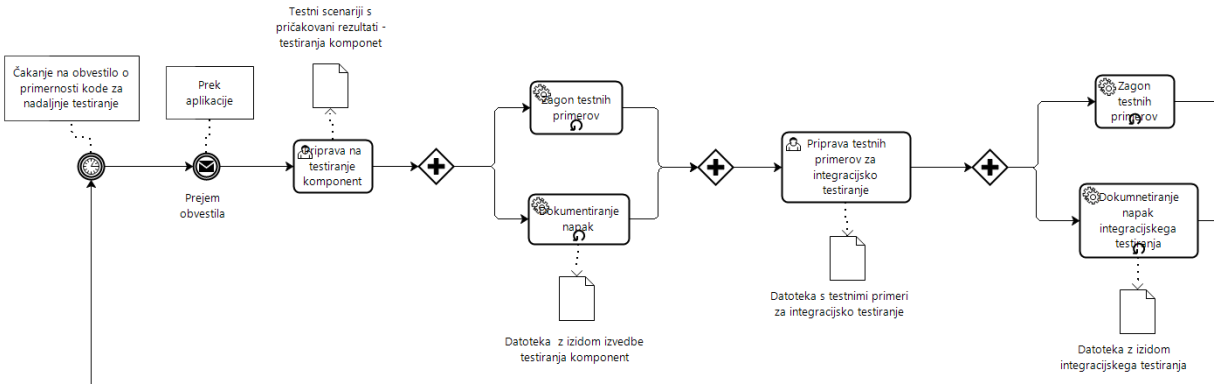
Slika 35: Predlog prenovljenega procesa – pregled kode



Vir: lastno delo.

Po prejemu obvestila, da je koda funkcionalnosti primerna za nadaljnje testiranje, razvijalec 1 na vsaki funkcionalnosti obvezno opravi obsežnejše testiranje komponent in integracijsko testiranje. V prvem koraku pripravi datoteko, ki vsebuje testne scenarije s pričakovanimi rezultati za testiranje komponent (ang. unittest). Zažene jih na kodi funkcionalnosti, program pa avtomatsko beleži napake. Po opravljenem testiranju komponent programer 1 opravi še integracijsko testiranje. Ponovno pripravi datoteko, ki vsebuje testne scenarije s pričakovanimi rezultati. Med izvajanjem integracijskega testiranja si sproti beleži napake. Na sliki 36 je prikazan del procesa, kjer se izvajajo aktivnosti, ki so potrebne za izvedbo faze testiranja komponent in izvedbo integracijskega testiranja.

Slika 36: Predlog prenovljenega procesa – izvedba testiranja komponent in integracijskega testiranja



Vir: lastno delo.

Za testiranjem komponent in integracijskim testiranjem sledi sistemsko testiranje. Za razliko od testiranja komponent in integracijskega testiranja je sistemsko testiranje obvezno le za določene funkcionalnosti.

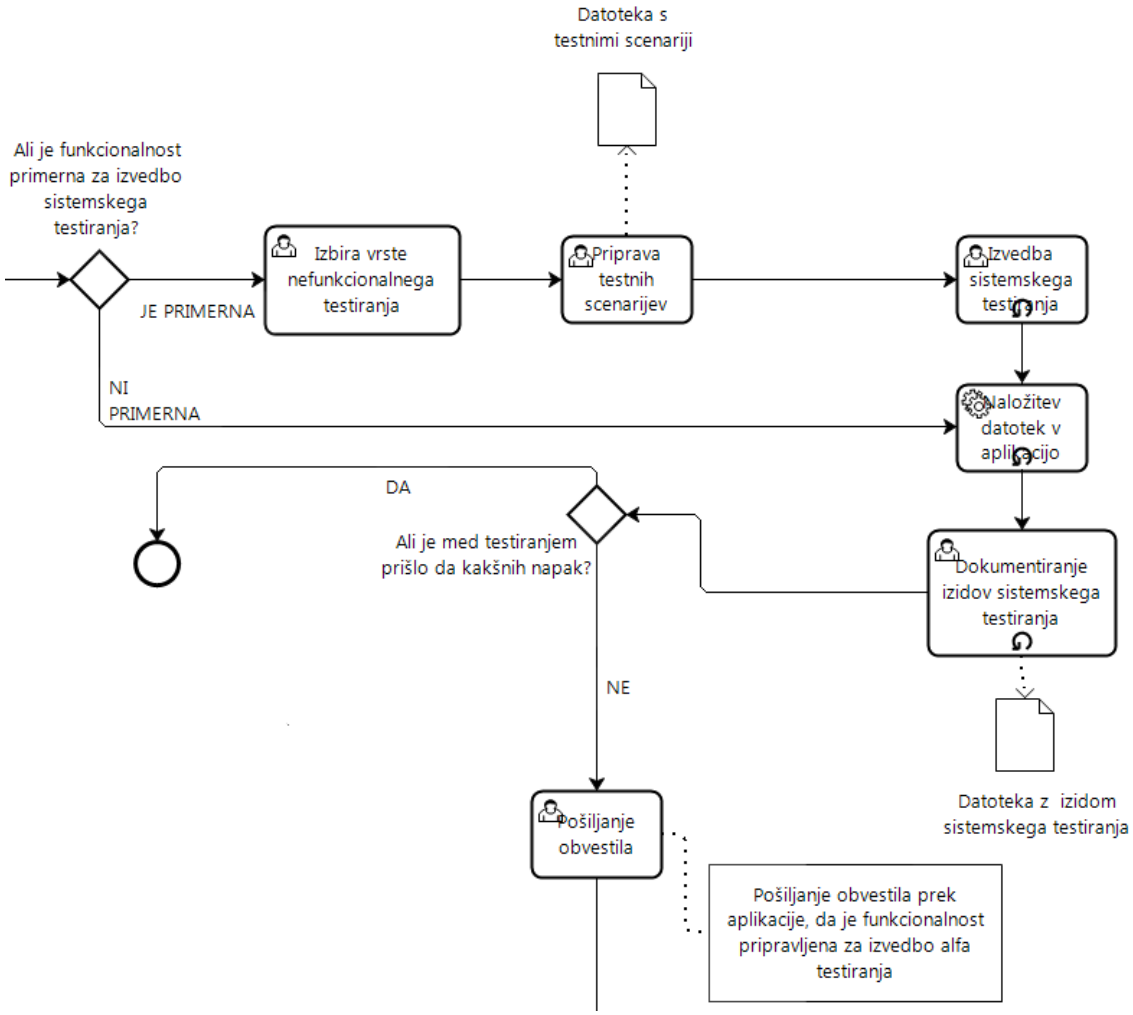
V primeru, da je koda funkcionalnosti pripravljena za izvedbo sistemskega testiranja, mora razvijalec 1 v aplikaciji izbrati vrste nefunkcionalnega testiranja, ki jih bo uporabil pri izvedbi sistemskega testiranja. Po pripravljenem načrtu za izvedbo sistemskega testiranja opravi testiranje in si zabeleži izid.

V primeru, ko razvijalec 1 v katerikoli fazi odkrije napake v delovanju funkcionalnosti, se proces nadaljnjega testiranja zaključi. Koda funkcionalnosti ponovno preide v fazo razvoja. Kljub temu, da koda funkcionalnosti ni primerna za nadaljnje testiranje, mora razvijalec 1 v aplikacijo naložiti vse datoteke z izidi vseh faz in vrst testiranj.

Ko bo v naslednji iteraciji izvajal testiranje, bo uporabil že napisane testne scenarije s pričakovanimi rezultati. Poleg tega pa bosta tudi tester in razvijalec 2 imela natančen vpogled, v kateri fazi testiranja se trenutno nahaja koda funkcionalnosti in kakšni testni scenariji so bili uporabljeni.

Na sliki 37 je prikazan del procesa, kjer se izvajajo aktivnosti, ki so potrebne za izvedbo faze sistemskega testiranja.

Slika 37: Predlog prenovljenega procesa – izvedba sistemskega testiranja



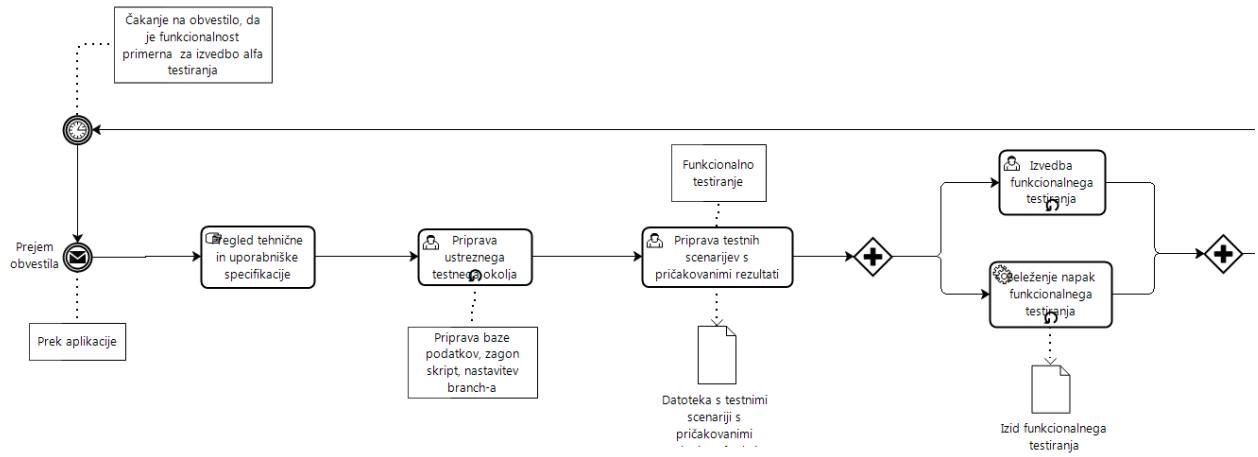
Vir: lastno delo.

V primeru, da razvijalec 1 v nobeni od faz ni našel napake, označi v aplikaciji, da je koda funkcionalnosti primerna za izvedbo alfa testiranja. Tester prek elektronske pošte dobi obvestilo, da lahko začne z izvajanjem alfa testiranja. V prvem koraku tester podrobneje preuči tehnično specifikacijo in uporabniška navodila. V drugem koraku si pripravi ustrezno testno okolje (tj. nastavi ustrezno bazo podatkov, prevzame novo verzijo aplikacije itd.). Pri prenovljenem procesu je alfa testiranje razdeljeno na funkcionalno in nefunkcionalno testiranje. Najprej tester izvede funkcionalno testiranje. V skladu z uporabniškimi navodili, funkcionalno specifikacijo in z vsemi informacijami, ki jih ima na voljo (od testiranja komponent, integracijskega ter sistemskega testiranja), si pripravi nabor testnih scenarijev s pričakovanimi rezultati.

Ko ima pripravljene testne scenarije, izvede funkcionalno testiranje. Hkrati s preizkušanjem delovanja funkcionalnosti v aplikaciji si sproti beleži izide testnih scenarijev. Na sliki 38 je prikazan del procesa, kjer se izvajajo aktivnosti, ki so potrebne za pripravo testnega okolja (priprava testne baze in zagon ustrezne kode programske rešitve

(ang. branch)) ter izvedbo faze funkcionalnega testiranja.

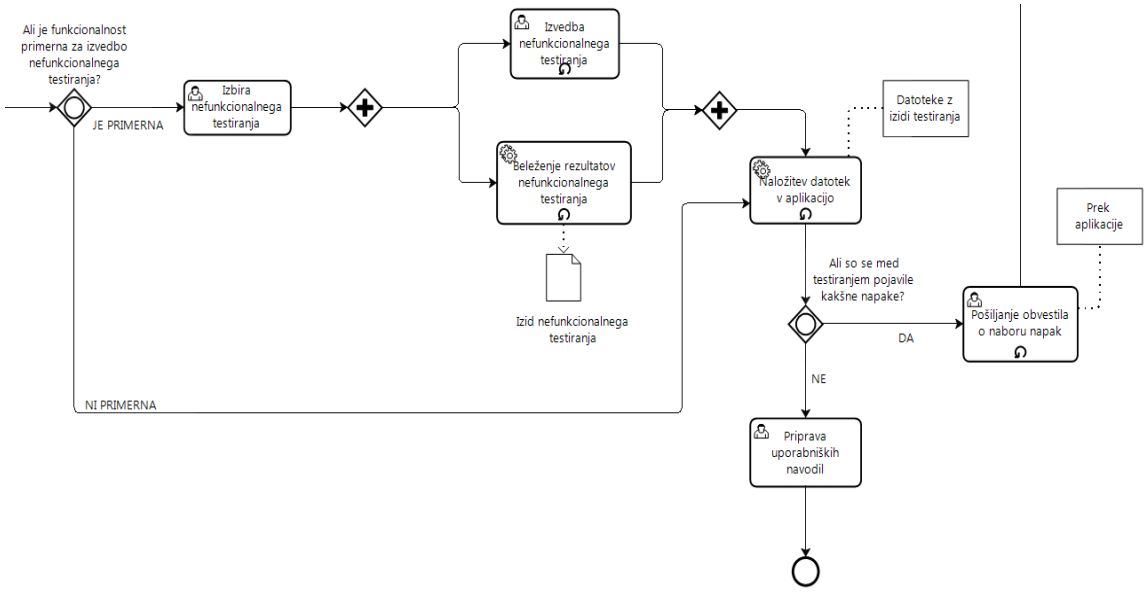
Slika 38: Predlog prenovljenega procesa – priprava testnega okolja in izvedba funkcionalnega testiranja



Vir: lastno delo.

Po zaključenem funkcionalnem testiranju sledi odločitev, ali je funkcionalnost primerna za izvedbo nefunkcionalnega testiranja. Če se izkaže, da je primerna, tester v aplikaciji izbere vrste nefunkcionalnega testiranja, ki jih bo uporabil. Med izvajanjem nefunkcionalnega testiranja si beleži rezultate testiranja. Na sliki 39 je prikazan del procesa, kjer se izvajajo aktivnosti, ki so potrebne za izvedbo nefunkcionalnega testiranja in dokumentiranje izida celotnega alfa testiranja.

Slika 39: Predlog prenovljenega procesa – izvedba nefunkcionalnega testiranja in dokumentiranje izida alfa testiranja



Vir: lastno delo.

Če po koncu izvedbe alfa testiranja, ki ga izvede izključno tester (lahko opravi funkcionalno in nefunkcionalno testiranje ali pa samo funkcionalno testiranje) ni bilo napak v delovanju funkcionalnosti, potem tester tudi pripravi uporabniška navodila za naročnika. S tem se celotni proces testiranja funkcionalnosti v izbranem podjetju zaključi. Funkcionalnost je pripravljena za predajo naročniku.

V primeru, da je tester med alfa testiranjem odkril napake, v aplikaciji označi, da so potrebni popravki. Razvijalec 1 dobi prek elektronske pošte obvestilo, da je bilo alfa testiranje neuspešno in da so zahtevani popravki. Tester v aplikacijo priloži datoteke s testnimi scenariji, ne glede na to, ali je bilo alfa testiranje uspešno ali neuspešno zaključeno.

3.9 Analiza AS–IS in TO–BE modela

V tem razdelku so predstavljeni rezultati analize AS–IS modela in TO–BE modela, in sicer časovna analiza AS–IS in TO–BE modela, stroškovna analiza AS–IS in TO–BE modela in obremenjenost virov.

3.9.1 Časovna analiza AS–IS modela

Čas trajanja vsake aktivnosti v procesu sem določila s pomočjo informacij, ki sem jih pridobila v izbranem podjetju. V podjetju namreč trenutno nikjer ne beležijo časa izvajanja posameznih aktivnosti v procesu. Na razpolago sem imela le podatek o trajanju posamezne stopnje življenjskega cikla programske rešitve. Zato sem po pogovoru z zaposlenimi in na podlagi lastnih delovnih izkušenj kot testerka programskih rešitev ocenila čas trajanja izvedbe celotnega procesa testiranja ter čas trajanja posamezne aktivnosti v procesu.

Kot že omenjeno, se v povprečju izvedejo tri do štiri iteracije procesa testiranja za eno obsežnejšo funkcionalnost ali več enostavnejših funkcionalnosti, preden se jih preda naročniku. Časovno analizo obstoječega procesa sem izvedla za štiri iteracije.

Celotni čas izvajanja procesa testiranja v štirih iteracijah je bil 3 dni 5 ur 11 min, kar je 77 ur 11 minut. V izbranem podjetju zaposleni v povprečju delajo osem ur na dan, izjemoma več. Če pretvorimo trajanje testiranja v delovne dneve, ugotovimo, da je za proces testiranja obsežnejše funkcionalnosti ali več enostavnejših potrebno 9 delovnih dni, 5 ur in 11 minut.

Podrobnejši rezultati časovne analize obstoječega procesa so v prilogi 2.

3.9.2 Časovna analiza TO–BE modela

V izbranem podjetju so pri prenovi procesa testiranja določili, da bi v okviru testiranja ene zahtevnejše funkcionalnosti ali pa več enostavnejših funkcionalnosti izvedli največ dve

iteraciji procesa testiranja. V redkih oziroma izjemnih primerih bi izvedli tri iteracije.

Predpogoj za izvedbo samo dveh iteracij testiranj je, da se na proces testiranja bolje pripravita razvijalec 1 in tester programskih rešitev. V razdelku, kjer sem podrobneje opisala predlog za prenovljeni proces, sem natančno opredelila nov postopek testiranja, ki poteka prek različnih faz testiranja.

Celotni čas izvajanja procesa testiranja v dveh iteracijah je bil 2 dni 9 ur 54 min, kar je 33 ur 54 min. Če pretvorimo trajanje procesa testiranja ene obsežnejše funkcionalnosti ali pa več enostavnejših v delovne dneve, je za testiranje potrebno 7 delovnih dni, 1 ura in 54 minut.

Podrobnejši rezultati za časovno analizo prenovljenega procesa so v prilogi 3.

3.9.3 Analiza virov AS–IS modela

Pri analizi virov sem se osredotočila na stroškovno analizo in obremenjenost virov, uporabljenih v procesu. V našem primeru so bili vir zaposleni. Zanimalo me je, koliko znaša strošek testiranja obsežnejše in zahtevnejše funkcionalnosti ali pa nabora več enostavnejših ter manj obsežnih funkcionalnosti in koliko so pri tem obremenjeni posamezni zaposleni, udeleženi v proces testiranja. Razpolagala sem s podatkoma, da se v izbranem podjetju večinoma izvede štiri iteracije procesa testiranja, preden se nadgradnjo programske rešitve preda naročniku. Časovna zahtevnost štirih iteracij procesa testiranja je – kot smo ugotovili v enem od prejšnjih podpoglavij – v povprečju 9 delovnih dni 5 ur in 11 minut. Pomembna informacija za izbrano podjetje je, kolikšni stroški nastanejo s testiranjem funkcionalnosti.

Za analizo stroškov izvedbe štirih iteracij procesa testiranja potrebujemo informacijo o plačah zaposlenih in koliko znašajo fiksni mesečni stroški na zaposlenega. Ker so informacije o plačah zaposlenih v izbranem podjetju poslovna skrivnost, sem na spletni strani www.placa.si pridobila podatek o povprečni bruto plači v Sloveniji za delovno mesto, ki ga zasedajo udeleženci poslovnega procesa.

Glede na delo, ki ga opravljajo v izbranem podjetju, so po klasifikaciji poklicev spletne strani www.placa.si razvrščeni po naslednjem ključu: razvijalec 1 je C# programer, razvijalec 2 je arhitekt programskih rešitev, tester programskih rešitev pa je tehnični inženir.

V tabeli 2 so podatki o delovnem mestu, ki ga zasedajo udeleženci procesa testiranja in o povprečni bruto plači v Sloveniji glede na njihovo delovno mesto.

Tabela 2: Podatki o delovnem mestu, ki ga zasedajo udeleženci procesa testiranja in o povprečni bruto plači v Sloveniji glede na njihovo delovno mesto

Delovno mesto	**Naziv delovnega mesta**	**Povprečna bruto plača v Sloveniji (v EUR)**
C# programer	razvijalec 1	2.137
Arhitekt programskih rešitev	razvijalec 2	3.114
Tehnični inženir	tester programskih rešitev	1.840

Vir: Placa.si, (2018).

Končni strošek delodajalca v Sloveniji je bruto bruto plača, kar je bruto plača zaposlenega, povečana za dodatke in prispevke delodajalca (16,1 odstotka) ter za povračilo stroškov v zvezi s delom (prevoz, prehrana, dodatki itd.).

Za potrebe magistrskega dela sem kot bruto bruto plačo upoštevala le bruto plačo zaposlenega, povečano za 16,1 odstotka. V tabeli 3 so vrednosti bruto bruto plače za posamezno delovno mesto, ki ga zasedajo udeleženci procesa testiranja.

Tabela 3: Vrednosti bruto bruto plače za posamezno delovno mesto, ki ga zasedajo udeleženci procesa testiranja

Delovno mesto	**Naziv delovnega mesta**	**Bruto bruto plača (v EUR)**
C# programer	razvijalec 1	2.481
Arhitekt programskih rešitev	razvijalec 2	3.615
Tehnični inženir	tester programskih rešitev	2.136

Vir: lastno delo.

V izbranem podjetju sem pridobila informacijo, da mesečni fiksni stroški na zaposlenega v povprečju znašajo okoli 300 EUR.

Strošek plače zaposlenega na uro in fiksni strošek na uro sem izračunala kot količnik med bruto bruto plačo zaposlenega in številom delovnih ur, ki jih zaposleni povprečno opravijo v enem mesecu. Pri izračunu sem upoštevala, da zaposleni v izbranem podjetju v povprečju opravijo 176 ur mesečno.

Fiksni strošek na uro je za vse udeležence procesa enak, in sicer 1,7 EUR. Bruto bruto strošek delodajalca za razvijalca 1 znaša 14 EUR, za razvijalca 2 21 EUR in za testerja 12 EUR na uro (zneski so zaokroženi). V tabeli 4 so prikazani fiksni stroški na uro za vsakega udeleženca posebej in urne postavke (tj. bruto bruto strošek delodajalca, ki mu je prištet še fiksni strošek) vsakega udeleženca v procesu testiranja. Opredelitev stroškov posameznega zaposlenega na uro je ključnega pomena za stroškovno analizo.

Tabela 4: Bruto bruto strošek delodajalca na uro glede na delovno mesto udeleženca v procesu in vrednost fiksnih stroškov

Zaposleni v procesu	**Bruto bruto strošek delodajalca (v EUR)**	**Fiksni strošek na uro (v EUR)**
razvijalec 1	14	1,7
razvijalec 2	21	1,7
tester programskih rešitev	12	1,7

Vir: lastno delo.

Kot rezultat stroškovne analize procesa testiranja, ki se je izvedel v štirih iteracijah, so v tabeli 5 prikazani naslednji stroški: fiksni strošek na posameznega zaposlenega in vsota fiksnih stroškov vseh treh zaposlenih; stroški glede na bruto urno postavko zaposlenega in vsota stroškov za vse tri zaposlene; celotni stroški kot vsota fiksnih stroškov in stroškov glede na bruto urne postavke posameznega zaposlenega ter vsota vseh stroškov procesa, ki se je izvedel v štirih iteracijah).

Tabela 5: Stroškovna analiza obstoječega procesa testiranja, izvedenega v štirih iteracijah

Delovno mesto zaposlenega	**Fiksni stroški na zaposlenega (v EUR)**	**Stroški glede na bruto urno postavko zaposlenega (v EUR)**	**Celotni stroški (v EUR)**
Razvijalec 1	21	175	196
Razvijalec 2	48	588	636
Tester programskih rešitev	62	432	494
Vsota stroškov po stolpcih (v EUR)	131	1.195	1.326

Vir: lastno delo.

Celotni stroški obstoječega procesa testiranja, ki je izveden v štirih iteracijah, znašajo 1.326 EUR.

Pomemben podatek, ki ga dobimo pri analizi virov, je podatek o izkoriščenosti posameznega vira v procesu testiranja.

V tabeli 6 so prikazani podatki o izkoriščenosti posameznega zaposlenega v obstoječem procesu testiranja (njihova skupna vsota izkoriščenosti v procesu je 100 odstotkov).

Tabela 6: Izkoriščenosti posameznega zaposlenega v obstoječem procesu testiranja

Delovno mesto zaposlenega	**Izkoriščenost zaposlenega v trenutnem procesu testiranja (v odstotkih)**
Razvijalec 1	25
Razvijalec 2	28
Tester programskih rešitev	47
Skupaj	100

Vir: lastno delo.

Trenutna izkoriščenost zaposlenih v proces testiranja ni optimalna. V obstoječem procesu testiranja v izbranem podjetju se povprečno izvedejo štiri iteracije procesa testiranja, preden gre programska rešitev k naročniku. Razvijalec 1, ki bi moral opraviti pomembnejši del testiranja, je v obstoječem procesu premalo obremenjen, saj je njegova izkoriščenost v procesu zgolj 25 odstotkov. Razvijalec 2, ki opravi pregled kode, je v po drugi strani preveč obremenjen, saj mora v povprečju kar štirikrat pregledati kodo programa za razvijalcem 1. To je za razvijalca 2, ki ni ključni udeleženec v procesu, prevelika obremenitev.

Tester je v obstoječem procesu premalo obremenjen, sploh glede na to, da je v izbranem podjetju njegovo ključno delo testiranje programskih rešitev. Trenutna skupna obremenitev razvijalca 1 in testerja v procesu znaša zgolj 43 odstotkov, kar je tudi posledica slabo zastavljenega procesa testiranja v izbranem podjetju.

3.9.4 Analiza virov TO–BE modela

Po enakem postopku kot sem naredila stroškovno analizo za AS–IS model, sem naredila tudi stroškovno analizo za TO–BE model, torej za model prenovljenega poslovnega procesa.

Plače zaposlenih, fiksni stroški na zaposlenega in povprečni mesečni delovni čas ostajajo pri prenovljenem procesu enaki.

Kot rezultat stroškovne analize za prenovljeni proces testiranja, ki se izvede v dveh iteracijah, so v tabeli 7 prikazani naslednji stroški: fiksni stroški na posameznega zaposlenega in vsota fiksnih stroškov vseh treh zaposlenih; bruto bruto strošek delodajalca za zaposlenega in vsota stroškov za vse tri zaposlene; celotni stroški kot vsota fiksnih stroškov in stroškov glede na bruto urno postavko posameznega zaposlenega ter vsota vseh stroškov procesa, ki se je izvedel v dveh iteracijah.

Tabela 7: Stroškovna analiza za prenovljeni proces testiranja, ki se izvede v dveh iteracijah

Delovno mesto zaposlenega	**Fiksni stroški na zaposlenega (v EUR)**	**Bruto bruto strošek delodajalca za zaposlenega (v EUR)**	**Celotni stroški (v EUR)**
Razvijalec 1	31	252	283
Razvijalec 2	25	315	340
Tester programskih rešitev	44	300	344
Vsota stroškov po stolpcih (v EUR)	100	867	967

Vir: lastno delo.

Celotni stroški prenovljenega procesa testiranja, ki je izveden v dveh iteracijah, znašajo 967 EUR. To je kar 27–odstotni prihranek v primerjavi s stroški procesa, izvedenega v štirih iteracijah.

V tabeli 8 so podatki o izkoriščenosti posameznega zaposlenega v prenovljenem procesu testiranja.

Tabela 8: Izkoriščenost zaposlenega v trenutnem procesu testiranja, v odstotkih

Delovno mesto zaposlenega	**Izkoriščenost zaposlenega v trenutnem procesu testiranja (v odstotkih)**
Razvijalec 1	32
Razvijalec 2	9
Tester programskih rešitev	59
Skupaj	100

Vir: lastno delo.

Izkoriščenosti zaposlenih v prenovljenem procesu je glede na vloge zaposlenih precej bolj smiselno razporejena v primerjavi z obstoječim procesom izbranega podjetja. Glede na to, da pomembnejši del testiranja v podjetju opravita razvijalec 1 in tester, je smiselno, da sta v procesu najbolj izkoriščena. Njuna skupna izkoriščenost v prenovljenem procesu znaša 91 odstotkov. Izkoriščenost razvijalca 1 in testerja je v prenovljenem procesu za kar 19 odstotkov večja kot v obstoječem procesu. Po pravilni implementaciji prenovljenega procesa v izbrano podjetje bi se izboljšala tudi produktivnost zaposlenih v procesu, njihov izkoristek pa bi bil bolj smiseln kot je v obstoječem procesu.

Glede na razporeditev virov pa bi se ustrezno zmanjšali tudi stroški procesa testiranja.

3.9.5 Analiza koledarja

Četrta stopnja analize procesa trenutno ni pomembna za obravnavano analizo, saj v izbranem podjetju ni točno določenega urnika začetka in konca dela, niti ni predpisano,

kdaj je čas malice. Od zaposlenih se zahteva le osemurna prisotnost vsak delovni dan.

Za izboljšanje obstoječega procesa so pomembni predvsem rezultati, ki sem jih pridobila s pomočjo časovne analize in analize virov.

3.9.6 Primerjava rezultatov stroškovne in časovne analize med AS–IS in TO–BE modelom za različno število testiranih funkcionalnosti

Za primerjavo stroškovne in časovne analize AS–IS in TO–BE modela sem kot izhodiščni podatek vzela število uspešno testiranih funkcionalnosti. Pri izračunu časovne in stroškovne zahtevnosti AS–IS in TO–BE modela za 5, 10, 50 in 100 testiranih funkcionalnosti sem upoštevala rezultate časovne in stroškovne analize, dobljene pri izračunu za eno testirano funkcionalnost (ki vključuje eno zahtevnejšo ali več enostavnejših funkcionalnosti)

V tabeli 9 so rezultati časovne in stroškovne analize AS–IS in TO–BE modela za različno število testiranih funkcionalnosti.

Tabela 9: Rezultati časovne in stroškovne analize AS–IS in TO–BE modela za različno število testiranih funkcionalnosti

Število funkcionalnosti	**Celotni čas izvedbe AS–IS modela**	**Celotni stroški izvajanja AS–IS modela (v EUR)**	**Celotni čas izvedbe TO–BE modela**	**Celotni stroški izvajanja TO–BE modela (v EUR)**
1	9 dni 5 ur 11 min	1.326	7 dni 1 ura 54 min	967
5	48 dni 1 ura 56 min	6.630	36 dni 1 ura 31 min	4.835
10	96 dni 3 ure 53 min	13.260	72 dni 3 ure 3 min	9.670
50	482 dni 3 ure 25 min	66.300	361 dni 7 ur 16 min	48.350
100	964 dni 6ur 51 min	132.600	723 dni 6 ur 33 min	96.700

Vir: lastno delo.

V tabeli 10 so časovni in stroškovni prihranki pri prenovljenem procesu testiranja za 1, 5, 10, 50 in 100 pretestiranih funkcionalnosti, in sicer v primerjavi z obstoječim procesom testiranja izbranega podjetja.

Tabela 10: Časovni in stroškovni prihranki pri prenovljenem procesu testiranja

Število funkcionalnosti	Časovni prihranek pri prenovljenem procesu	Stroškovni prihranek pri prenovljenem procesu (v EUR)	Stroškovni prihranek pri prenovljenem procesu (v odstotkih)
1	2 dni 3 ure 17 min	359	27
5	12 dni 25 min	1.795	27
10	24 dni 50 min	3.590	27
50	120 dni 20 ur 9 min	17.950	27
100	241 dni 18 min	35.900	27

Vir: lastno delo.

S primerjavo stroškovne in časovne analize AS–IS in TO–BE modela sem dokazala, da je časovna izvedba TO–BE modela krajša in stroškovno učinkovitejša. To velja tako pri manjšem kot pri večjem številu testiranih funkcionalnosti.

4 APLIKACIJA ZA PODPORO PRENOVLJENEMU PROCESU

Aplikacija, ki jo bom predstavila v nadaljevanju, je podpora prenovljenemu procesu testiranja. Izdelala sem jo samostojno na podlagi predloga prenovljenega procesa testiranja programskih rešitev, in sicer v Bizagi Studio. V aplikacijo sem vključila ključne korake procesa testiranja, ki so implementirani v aplikacijo.

Za izgradnjo aplikacije sem potrebovala podatkovni model, ki je predstavljen v prilogi 1.

Aplikacija je namenjena uporabi vsem zaposlenim, ki so vključeni v proces testiranja in ki želijo imeti celovit pregled nad izidom testiranja vsake faze posebej.

Razvijalec, ki razvije programsko rešitev za določeno funkcionalnost, je zadolžen, da v prenovljenem procesu opravi naslednje faze testiranja: testiranje komponent, integracijsko testiranje in sistemsko testiranje. Zaradi lažje izvedbe alfa testiranja je razvijalec zadolžen, da za vsako izvedeno fazo dokumentira izid testiranja in v aplikacijo naloži datoteko, ki vsebuje testne scenarije s pričakovanimi rezultati.

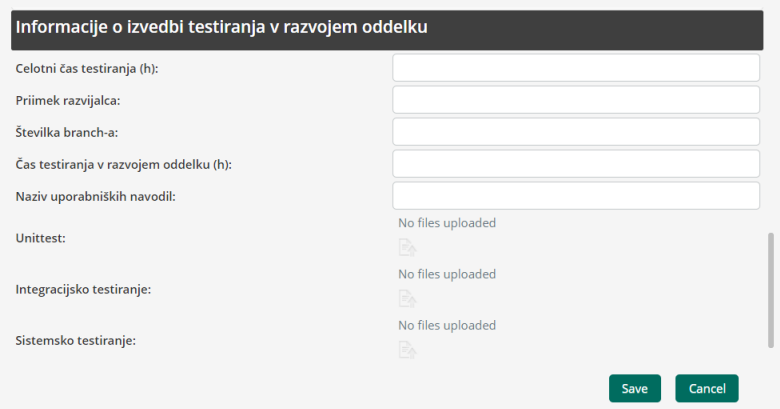
V aplikaciji ima razvijalec formo Informacije o izvedbi testiranja v razvojnem oddelku, ki je prikazana na sliki 40. Forma vsebuje naslednja vnosna polja:

- Celotni čas testiranja funkcionalnosti (h): v vnosno polje se vnese celotni čas testiranja funkcionalnosti (v urah);
- Priimek razvijalca: razvijalec v vnosno polje vnese svoj priimek;
- Številka branch–a: razvijalec vnese verzijo aplikacije, v kateri se nahaja programska koda za novo funkcionalnost;
- Čas testiranja v razvojnem oddelku (h): razvijalec vnese celotni čas testiranja (v urah), ki ga je porabil za vse tri faze testiranja (testiranje komponent, integracijsko testiranje

in sistemsko testiranje). Podatek je pomemben, saj se ob koncu testiranja preveri, ali se je testiranje v razvojem oddelku izvedlo v časovnem okvirju, predvidenem na začetku življenjskega cikla razvoja programske rešitve;
- Naziv uporabniških navodil: razvijalec vnese naziv uporabniških navodil, ki jih je sestavil tester funkcionalnosti in v katerih je opis delovanja funkcionalnosti;
- Unittest: razvijalec v aplikacijo naloži datoteko s testnimi scenariji in pričakovanimi rezultati za testiranje komponent. Enako naredi za vnos datoteke pri integracijskem in sistemskem testiranju;
- Integracijsko testiranje;
- Sistemsko testiranje.

S klikom na gumb »Save«, ki je spodaj desno, se bo v aplikaciji ustvaril nov ID zapisa funkcionalnosti, ki predstavlja zaporedno številko testirane funkcionalnosti.

Slika 40: Forma – Informacije o izvedbi testiranja v razvojnem oddelku



Vir: lastno delo.

Ko razvijalec zaključi s testiranjem funkcionalnosti, začne s testiranjem tester. Tako kot razvijalec mora tudi tester dokumentirati izide testiranja.

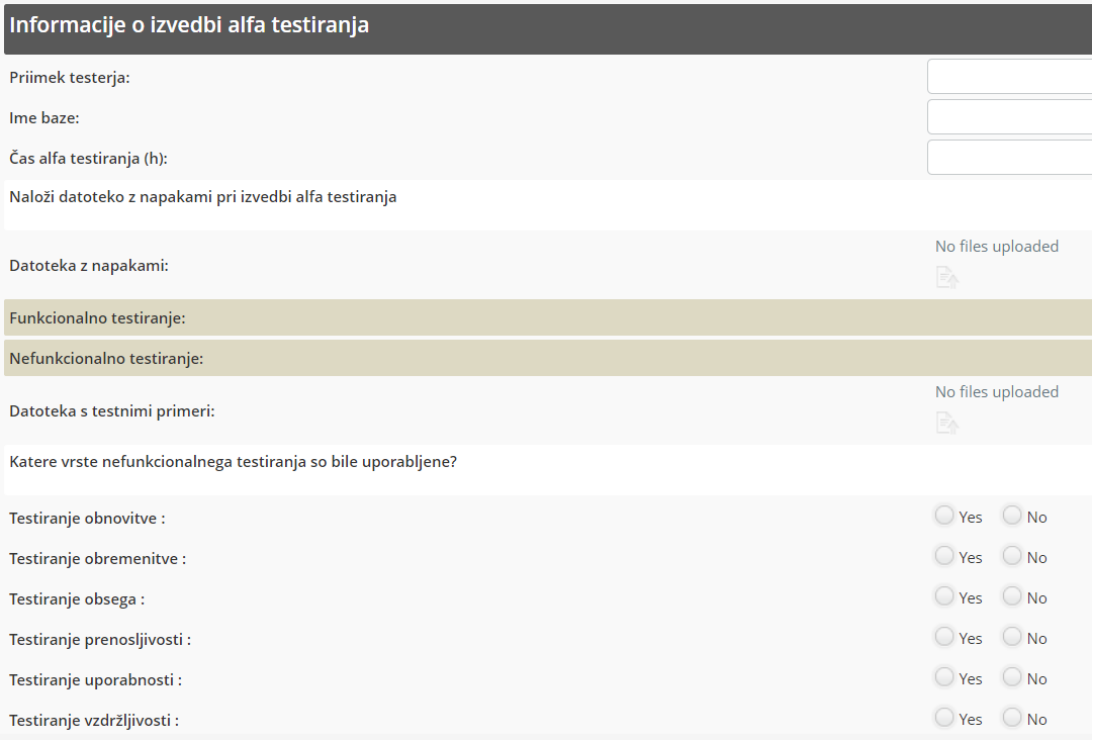
V aplikaciji ima tester formo »Informacije o izvedbi alfa testiranja«, ki je prikazana na sliki 41. Forma vsebuje naslednja vnosna polja:

- Priimek testerja: tester v vnosno polje vnese svoj priimek;
- Ime baze: tester vnese ime testne baze, ki jo je uporabil v fazi alfa testiranja;
- Čas alfa testiranja (h): tester vnese celotni čas testiranja (v urah), ki ga je porabil za funkcionalno in nefunkcionalno testiranje v okviru alfa testiranja. Podatek je pomemben, saj se na koncu testiranja preveri, ali se je alfa testiranje izvedlo časovnem okvirju, predvidenem na začetku življenjskega cikla razvoja programske rešitve;
- Datoteka z napakami: tester v aplikacijo naloži datoteko z napakami, ki jih je našel v eni iteraciji celotnega alfa testiranja;
- Datoteka s testnimi primeri: tester naloži datoteko s testnimi primeri in pričakovanimi

rezultati za celotno alfa testiranje;

– Izbira nefunkcionalnega testiranja: ker za vsako funkcionalnost niso primerne vse vrste nefunkcionalnega testiranja, mora tester označiti tiste vrste nefunkcionalnega testiranja, ki jih je uporabil.

Slika 41: Forma – Informacije o izvedbi alfa testiranja



Vir: lastno delo.

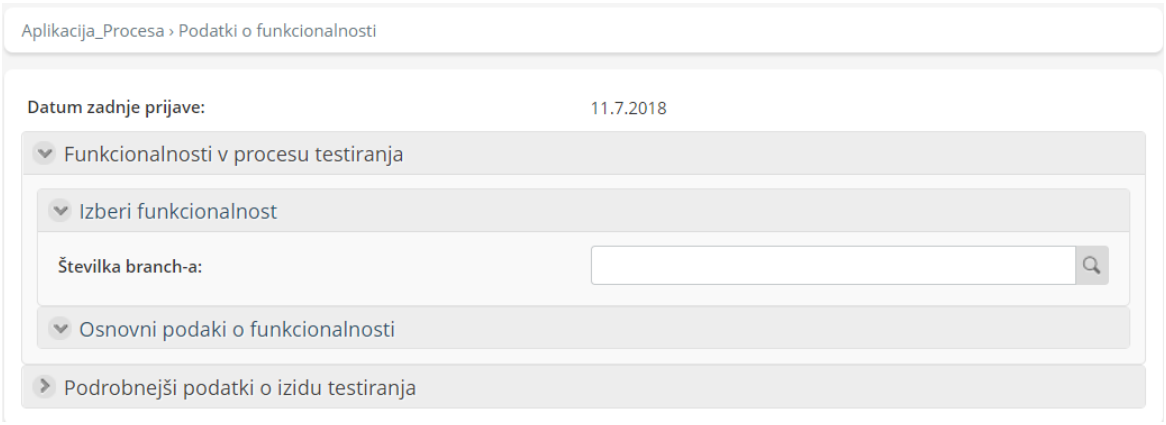
Za vsako funkcionalnost posebej se v aplikacijo shranjujejo izidi vseh faz in vrst testiranja. Namen ustvarjene aplikacije pa ni samo pregled izidov testiranja za vsako funkcionalnost posebej, temveč da se testne scenarije s pričakovanimi rezultati, ki so bili uporabljeni pri določeni funkcionalnosti, lahko uporabi pri testiranju podobnih funkcionalnosti. Primer: uvozi in izvozi datotek iz aplikacije za različna poročila so v določeni aplikaciji implementirani na podoben način, zato testne scenarije za uvoze in izvoze različnih datotek le prilagodimo glede na obstoječe že uporabljene testne scenarije, ki bodo na voljo v aplikaciji, ki sem jo naredila.

Ko bo v aplikacijo vnesenih nekaj izidov testiranja – bodisi za enako funkcionalnost ali pa različne funkcionalnosti – si bodo lahko zaposleni ogledali izid testiranja za določeno funkcionalnost.

Na vstopni strani aplikacije, ki je prikazana na sliki 42, se bo uporabniku izpisal datum zadnje prijave v aplikacijo. Na voljo bo imel možnost vnosa številke branch–a, s pomočjo katere bo našel izid testiranja za določeno funkcionalnost. Za vsako funkcionalnost posebej si bo uporabnik lahko ogledal osnovne podatke o tej funkcionalnosti in podrobnejše

podatke o celotnem izidu testiranja.

Slika 42: Vstopna stran aplikacije



Vir: lastno delo.

V nadaljevanju sta opisana dva primera uporabe aplikacije.

Prvi primer prikazuje vnos izida vseh faz in vrst testiranja za določeno funkcionalnost.

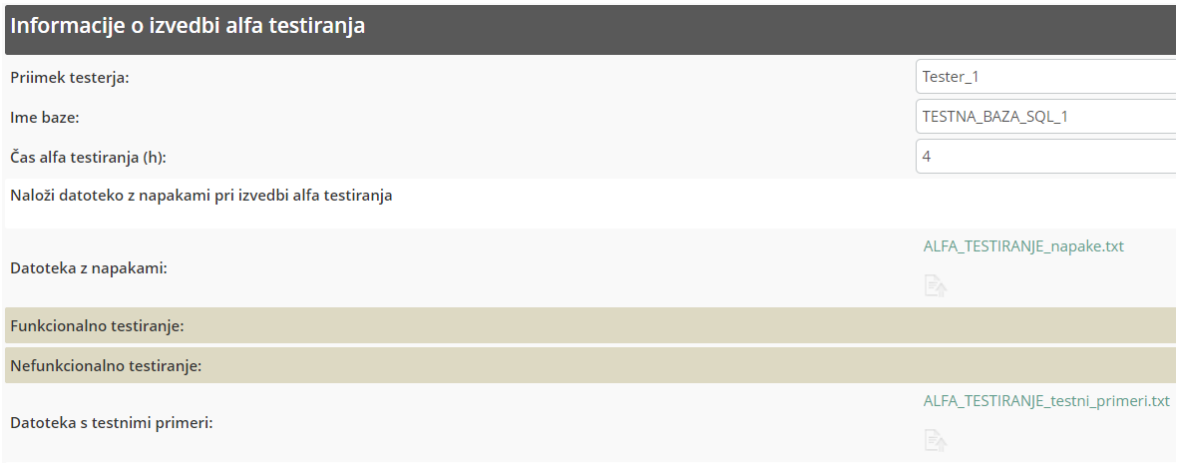
Za želeno funkcionalnost vnesemo podatke o izvedbi testiranja v razvojnem oddelku in o izvedbi alfa testiranja. Primer vnosa podatkov o izvedbi testiranja v razvojnem oddelku je prikazan na sliki 43.

Slika 43: Primer vnosa podatkov o izvedbi testiranja v razvojnem oddelku

Vir: lastno delo.

Primer vnosa podatkov o izvedbi alfa testiranja je prikazan na sliki 44.

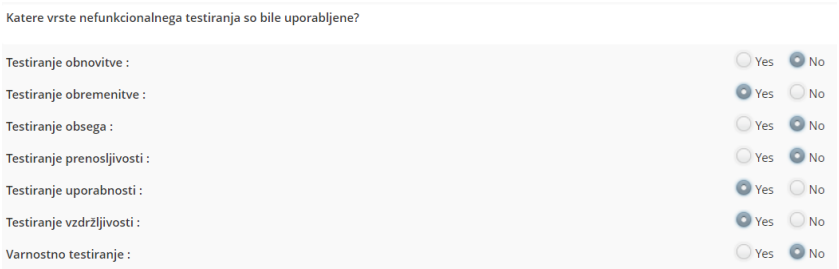
Slika 44: Primer vnosa podatkov o izvedbi alfa testiranja



Vir: lastno delo.

Na sliki 45 je primer prikaza, katere vrste nefunkcionalnega testiranja sem izbrala kot testni primer.

Slika 45: Primer izbire vrst nefunkcionalnega testiranja



Vir: lastno delo.

Na koncu shranimo nov vnos.

V drugem primeru želim poiskati izid testiranja za funkcionalnost, ki sem jo vnesla v prvem primeru. V iskalno polje vnesem številko branch–a, v mojem primeru 1.

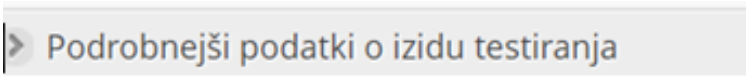
Pod zavihkom Osnovni podatki o funkcionalnosti, ki je prikazan na sliki 46, se izpišejo osnovni podatki o izidu testiranja.

Slika 46: Zavihek – Osnovni podatki o funkcionalnosti

Vir: lastno delo.

Če želi uporabnik podrobnejše informacije o izidu testiranja, klikne na zavihek Podrobnejši podatki o izidu testiranja, ki je prikazan na sliki 47.

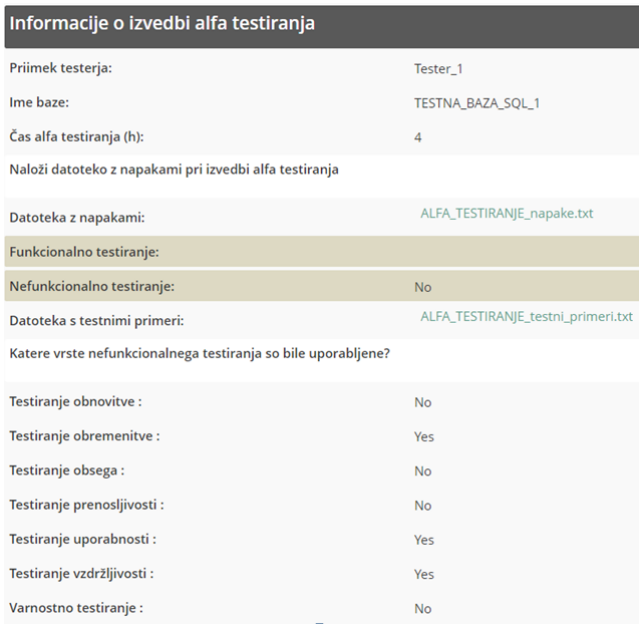
Slika 47: Zavihek – Podrobnejši podatki o izidu testiranja



Vir: lastno delo.

Uporabniku se prikažejo podrobnejše informacije o izidu testiranja za funkcionalnost, ki so prikazane za primer, za katerega sem vnesla celotni izid testiranja. Na sliki 48 so podrobnejše informacije o izvedbi alfa testiranja za uporabljen primer.

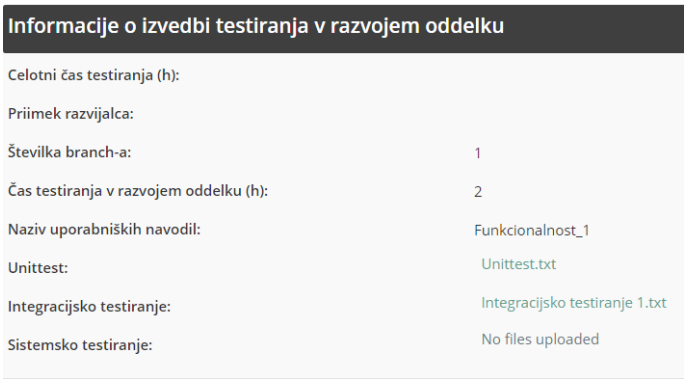
Slika 48: Uporabljen primer – Podrobnejše informacije o izvedbi alfa testiranja



Vir: lastno delo.

Na sliki 49 so podrobnejše informacije o izvedbi testiranja v razvojnem oddelku za uporabljen primer.

Slika 49: Uporabljen primer – Informacije o izvedbi testiranja v razvojnem oddelku



Vir: lastno delo.

4.1 Implementacija prenovljenega procesa v izbranem podjetju

V zadnjem poglavju magistrskega dela bom opisala, kako naj se v izbranem podjetju lotijo prenove oziroma implementacije prenovljenega procesa testiranja in ostalih temeljnih procesov. V opisu implementacije prenovljenega procesa testiranja v izbranem podjetju so uporabljena teoretična in praktična izhodišča, ki so bila podrobneje opisana v celotnem magistrskem delu.

V izbranem podjetju naj se prenove poslovanja lotijo sistematično in po korakih. Ker v izbranem podjetju načrtujejo prenovo in digitalizacijo vseh temeljnih procesov, naj pred prenovo poslovanja na novo opredelijo strategijo in vizijo podjetja ter na novo oblikujejo poslovni model. Pred izvedbo prenove naj določijo člane ekipe, ki bo izvedla prenovo in digitalizacijo temeljnih procesov.

Zaradi lažje orientacije in opredelitve podpornih in temeljnih procesov v podjetju naj kot osnovo vzamejo Porterjevo verigo vrednosti. Vse temeljne in podporne procese naj podrobneje opredelijo tako opisno kot v obliki procesnih diagramov s pomočjo izbranega orodja MPP.

Prenova in digitalizacija temeljnih poslovnih procesov naj poteka po zaporednih fazah življenjskega cikla managementa poslovnih procesov. V življenjski cikel managementa poslovnih procesov naj vključijo tri temeljne faze prenove in digitalizacije poslovanja s pomočjo informacijske tehnologije, kot sem opredelila v magistrskem delu.

Za lažjo oceno trenutnega celotnega življenjskega cikla razvoja programske opreme naj uporabijo model CMMI, ki ga uporabljajo vsa večja podjetja, katerih temeljna dejavnost je razvoj programske opreme. Z uporabo modela bodo v izbranem podjetju lahko uvrstili

celotni proces razvoja programske opreme na ustrezno zrelostno stopnjo. To bo izbranemu podjetju v pomoč, saj bo postalo jasno, kakšne pomanjkljivosti ima trenutni življenjski cikel razvoja programske opreme. V magistrskem delu sem na podlagi interne dokumentacije, s pomočjo pogovora z enim izmed projektnih vodji in arhitektom programskih rešitev, ki zelo dobro poznata obstoječi proces razvoja programske rešitve izbranega podjetja, in na podlagi lastnih delovnih izkušenj kot testerka programskih rešitev uvrstila življenjski cikel razvoja programske opreme na drugo zrelostno stopnjo.

Skupaj s projektnim vodjem in arhitektom programskih rešitev sem oblikovala predloge, ki bi jih moralo izbrano podjetje upoštevati, če bi želelo doseči tretjo zrelostno stopnjo razvoja programske rešitve, kjer so procesi standardizirani. Predlogi so naslednji:

- Uspešnost in učinkovitost poslovnih procesov bi v podjetju morali meriti vsaj kvalitativno.
- Natančneje je potrebno opredeliti vsak korak v procesu in določiti časovni okvir procesa razvoja ter procesa testiranja programske rešitve.
- Z natančnim popisom in opredelitvijo procesa bi v podjetju pridobili informacijo o učinkovitosti in uspešnosti izvajanja procesov.
- Proces uvajanja novih sodelavcev bi moral biti sistematično opredeljen na način, da bi ob prenosu znanja od obstoječih zaposlenih do novo zaposlenih ostali podprocesi v procesu razvoja programskih rešitev potekali kar se da nemoteno.
- Proces uvajanja novo zaposlenih je ključnega pomena, saj v izbranem podjetju želijo zagotoviti, da njihovi zaposleni opravijo svoje delo kakovostno in da je programska rešitev predana naročniku v dogovorjenem časovnem okvirju s čim manj napakami.
- Ker se procesi v izbranem podjetju med seboj prepletajo in dopolnjujejo (npr. proces testiranja programskih rešitev, ki se povezuje in dopolnjuje tudi s procesom razvoja programskih rešitev), bi bilo potrebno natančno definirati, kateri procesi se med seboj prepletajo in na kakšen način.
- Vsak zaposleni ima sicer določeno nalogo oziroma vlogo v procesu razvoja programske rešitve, a se med procesom vloge lahko tudi spremenijo (npr. inženir za zagotavljanje kakovosti programskih rešitev lahko testira programske rešitve na več projektih hkrati in ne samo na enem projektu, za katerega je bil določen na začetku procesa). Zato naj se vloge zaposlenih na začetku procesa razvoja programske rešitve natančno določijo in naj se njihove naloge med procesom drastično ne spreminjajo.
- Bolj natančno bi bilo potrebno določiti naloge in vlogo projektnega vodje v času procesa razvoja programske rešitve.

Izbranemu podjetju – v primeru, da se bodo odločili za prenovo poslovnih procesov ali če bodo želeli izboljšati procese in preiti na višjo zrelostno stopnjo – predlagam, da podrobneje preučijo CMMI in ga uporabijo pri prenovi poslovnih procesov.

Izbranemu podjetju predlagam, da se pri prenovi temeljnih poslovnih procesov ne osredotočajo zgolj na digitalizacijo poslovnih procesov, temveč naj v prenovo vključijo vse

socio–tehnične dejavnike iz Leavittovega diamanta. Le na ta način bodo namreč uspešno izvedli prenovo temeljnih poslovnih procesov. Vodstvo v izbranem podjetju naj upošteva, da zaposleni nimajo težav pri uporabi nove informacijske tehnologije, saj se pri vsakodnevnih delovnih nalogah srečujejo z uporabo novih programskih orodij in z veseljem nadgrajujejo svoje znanje s področja informacijske tehnologije. Ker je podjetje v manj kot enem letu popolnoma spremenilo organizacijo, zaposlilo približno dvajset novih sodelavcev in razširilo poslovanje, naj Leavittov diamant služi kot temelj pri prenovi poslovanja skozi celotni življenjski cikel managementa poslovnih procesov.

V magistrskem delu sem naredila predlog prenove in digitalizacije procesa testiranja programskih rešitev s pomočjo orodja MPP, in sicer programa Bizagi Modeler in Bizagi Studio. Spletna stran Finances Online reviews for business (2017) je leta 2017 uvrstila orodje Bizagi na osmo mesto med petnajstimi najboljšimi orodij MPP na svetu. S pomočjo njegove uporabe sem v magistrskem delu pokazala, da je to orodje, ki nudi vse potrebno za izvedbo prenove in digitalizacije procesa oziroma celotnega poslovanja podjetja.

Ker so v izbranem podjetju zaposleni strokovnjaki, ki so v preteklosti že razvijali aplikacije z različnimi orodij MPP, izbranemu podjetju predlagam, da pri izdelavi aplikacij s pomočjo npr. Bizagi BPM Suite vključijo zaposlene, ki imajo že izkušnje pri razvoju aplikacij z orodij MPP.

Pri izbiri orodja MPP naj v izbranem podjetju preučijo vse prednosti in pomanjkljivosti uporabe orodij MPP. Izbira ustreznega ponudnika orodja MPP mora biti skrbno premišljena. Ključne točke, po katerih naj podjetje izbere ponudnika orodja MPP, so opisane v magistrskem delu.

Če se bodo v izbranem podjetju pri prenovi temeljnih procesov odločili za uporabo orodja MPP, potem je ena izmed možnosti Bizagi BPM Suite. Ker podjetje Bizagi omogoča uporabnikom brezplačno uporabo Bizagi Modeler in Bizagi Suite, naj v izbranem podjetju sprva preučijo omenjeni orodji. V primeru, da se bodo v izbranem podjetju odločili za lastni razvoj aplikacij, v katere bodo implementirani temeljni poslovni procesi, ki bodo zajemali vse faze razvoja programske opreme, naj preučijo stroške lastnega razvoja aplikacije, ki jo bodo uporabljali zgolj zaposleni v podjetju, in stroške nakupa licenc. Ker bodo aplikacije uporabljali vsi zaposleni, ki so vključeni v temeljne poslovne procese, bodo v izbranem podjetju morali kupiti licence iz paketa Bizagi Engine, ki bo vsem zaposlenim omogočalo uporabo aplikacij na prenosnih računalnikih, tablicah ali pa mobilnih telefonih.

Ker so temeljni procesi kompleksni in vsebujejo veliko aktivnosti, izbranemu podjetju predlagam, da v aplikaciji Jira podrobneje opredeli vsako aktivnost v celotnem življenjskem ciklu razvoja programske opreme. A tudi v tem primeru bodo morali od podjetja Atlassian dokupiti dodatne funkcionalnosti v Jiri.

Stroške razvoja aplikacije s pomočjo programskega paketa Bizagi BPM Suite in zakup

določenega števila licenc ter nadgradnjo aplikacije Jira naj v izbranem podjetju temeljito preučijo z analizo koristi ter stroškov, ki jih prinaša omenjena izbira. Ker v izbranem podjetju trenutno ne znajo oceniti, koliko časa bi trajal razvoj interne aplikacije za podporo temeljnim procesom, ne morem opredeliti tovrstnih stroškov.

Pri prenovi procesa testiranja naj v izbranem podjetju upoštevajo faze in vrste testiranj, ki so opisane v tem magistrskem delu. Faze in vrste testiranja so prilagojene potrebam procesa testiranja, ki sem jih opredelila in podrobnejše razdelala s pomočjo analize obstoječega procesa testiranja, interne dokumentacije ter glede na obstoječo problematiko procesa. V razdelku, kjer je opredeljena podrobnejša časovna in stroškovna analiza obstoječega in prenovljenega procesa, lahko v izbranem podjetju že zelo kmalu po uvedbi novih faz in vrst testiranj in z dobro razdelanim procesom zmanjšajo stroške testiranja in čas izvedbe testiranja funkcionalnosti. Podrobnejši rezultati stroškovnega in časovnega prihranka prenovljenega procesa testiranja so predstavljeni v podpoglavju 3.9.6. V njem je predstavljena tudi primerjava rezultatov stroškovne in časovne analize med AS–IS in TO–BE modeloma za različno število testiranih funkcionalnosti.

SKLEP

V magistrskem delu sem pri prenovi in digitalizaciji procesa testiranja programskih rešitev, ki je eden temeljnih procesov v izbranem podjetju, uporabila Bizagi Modeler in Bizagi Studio iz programskega paketa Bizagi BPM Suite.

Z uporabo omenjenih orodij sem prikazala, kako uporabna so orodja MPP, saj jih lahko podjetja uporabijo na vsaki stopnji življenjska cikla managementa poslovnih procesov. Bizagi BPM Suite vsebuje inteligentno BPM platformo, ki uporabniku omogoča naslednje: modeliranje in analizo poslovnih procesov, razvoj samostojnih aplikacij, v katere so integrirani poslovni procesi, oblikovanje poslovnih pravil, uporabo rešitve v oblaku, vpogled v potek poslovnega procesa v realnem času ter zmožnost avtomatičnega izboljšanja in nadgrajevanja poslovnih procesov.

Na spletu je ogromno orodij MPP, zato naj se podjetja premišljeno odločijo za izbiro ustreznega. Orodje MPP morajo izbrati glede na potrebe in želje po prenovi poslovnih procesov oziroma poslovanja.

Preden se podjetje loti prenove in digitalizacije poslovanja, mora natančno opredeliti in analizirati obstoječe poslovne procese in opredeliti, kaj želi doseči s prenovo in digitalizacijo. Dandanes je pomembno, da podjetja sledijo spremembam sodobne informacijske tehnologije. Prenova poslovanja namreč ni mogoča brez napredne tehnologije. A podjetja se morajo kljub temu zavedati, da uspešne prenove in digitalizacije procesov ne bo mogoče izvesti, če ne bodo upoštevala vseh dejavnikov v podjetju, ki so opredeljeni v Leavittovem diamantu, torej tako sociooloških kot tehnoloških. V prenovo poslovanja bi morali biti obvezno vključeni vsi zaposleni, ki jih zadeva določen proces, saj

bodo le tako lažje in hitreje sprejeli spremembe.

Prenova in digitalizacija poslovanja je zahteven in dolgotrajen projekt. Rezultati pravilne implementacije prenovljenih procesov pa so v podjetju lahko vidni že zelo kmalu – lahko že v prvih treh mesecih po uvedbi prenovljenih procesov (Allsup, 2017).

Sodobna informacijska tehnologija torej igra pomembno vlogo pri prenovi in digitalizaciji poslovanja. A podjetja velikokrat pozabijo pred digitalizacijo in prenovo poslovnih procesov natančno opredeliti in analizirati predloge prenovljenih procesov.

V magistrskem delu sem pred izrisom procesnega diagrama predlaganega prenovljenega proces opisala, kako naj bi izgledal predlagani prenovljeni proces testiranja programskih rešitev, in se šele nato lotila izrisa diagrama ter razvoja aplikacije, ki nudi podporo prenovljenemu procesu testiranja.

Velikokrat podjetja zaradi pomanjkanja časa slabo dokumentirajo obstoječe in prenovljene procese, zato nikoli ne bodo mogla doseči učinkovitih in uspešnih procesov. Smiselno je, da se lotijo prenove in digitalizacije poslovanja na tak sistematičen način, kot je predstavljeno v magistrskem delu.

Podjetjem, katerih primarna dejavnost je razvoj, testiranje, implementacija, nadgradnja in vzdrževanje programskih rešitev, svetujem, da temeljito razdelajo vsako stopnjo oziroma podproces v življenjskem ciklu razvoja programskih rešitev. V primeru, ko se lotijo prenove ali pa kreiranje procesa testiranja programskih rešitev, naj preučijo že obstoječe procese testiranj, ki jih lahko najdejo na spletu in so sestavljeni iz različnih faz in vrst testiranj. Pomembno je, da se podjetja zavedajo, da je vsak proces testiranja svojevrsten in prilagojen glede na celotni življenjski cikel razvoja programskih rešitev in vrsto programske rešitve.

LITERATURA IN VIRI

1. Alwan. M. (2015, 9. januar). What is system development life cycle? *Airbrake*. Pridobljeno 15. junija 2018 iz https://airbrake.io/blog/sdlc/what-is-system-development-life-cycle
2. Appian. (2018). *Business Process Management (BPM) Articles*. Pridobljeno 15. januarja 2018 iz https://www.appian.com/bpm/business-process-management-articles/
3. Araujo, C. (2017, 6.marec). Why BPM is now taking a central role in digital transformation. *CIO FROM IDG*. Pridobljeno 20. novembra 2017 iz https://www.cio.com/article/3176077/software/why-bpm-is-now-taking-a-central-role-in-digital-transformation.html
4. Aryel Ramos, A. (2017. 12. januar). 7 curcial steps for choosing the right BPM software. *PEX PROCESS EXCELLENCE NETWORK*. Pridobljeno 10. aprila 2018 iz https://www.processexcellencenetwork.com/business-process-management-

bpm/news/7-crucial-steps-for-choosing-the-right-bpm
5. ATLASSIAN. (2018). *Jira Software*. Pridobljeno 3. septembra 2018 iz https://www.atlassian.com/software/jira
6. Attaran, M. (2003). Exploring the relationship between information technology and business process reengineering. *Information & Management, 41*(5), 1–25.
7. White, S.A. (2005, 10. marec). Using BPMN to model a BPEL process. *BPTrends.* Pridobljeno 10. novembra 2017 iz https://www.bptrends.com/publicationfiles/03-05%20WP%20Mapping%20BPMN%20to%20BPEL-%20White.pdf
8. Barjis, J. (2008). The importance of business process modeling in software systems design. *Science of Computer Programming, 71*(1), 73–87.
9. Beth Chrissis, M., Konrad, M. & Shrum, S. (2011). *CMMI for Development, Guidelines for process integration and product improvement*. Boston: Software engineering institute.
10. Bednarski, M. (2014, 4. julij). Case study: Multinational bank transforms its BPM program with a back – to bascis approach. *PEX Process excellence network*. Pridobljeno 28. septembra 2017 iz https://www.processexcellencenetwork.com/business-process-management-bpm/articles/case-study-multinational-bank-transforms-its-bpm-p
11. Bizagi. (2018). *Bizagi official web site*. Pridobljeno 10. septembra 2017 iz https://www.bizagi.com/
12. Bizagi Engine. (2018). *Bizagi Engine*. Pridobljeno 2. marca 2018 iz http://help.bizagi.com/bpm-suite/en/index.html?bizagiengine_jee.htm
13. Bizagi Modeler. (2018). *Bizagi Modeler*. Pridobljeno 20. februarja 2018 iz http://help.bizagi.com/process-modeler/en/index.html
14. Bizagi Studio. (2018). *Bizagi Studio*. Pridobljeno 7. marca 2018 iz http://help.bizagi.com/bpm-suite/en/index.html?process_execution.htm
15. BPMInstitute.org. (2018). *How to select the right tools for your BPM initiative*. Pridobljeno 23. avgusta 2017 iz http://www.bpminstitute.org/resources/articles/how-select-right-tools-your-bpm-initiative
16. Bright hub project management. (2017). *A look of the components of Leavitt's diamond.* Pridobljeno 21. decembra 2017 iz https://www.brighthubpm.com/change-management/122495-a-look-at-the-components-of-leavitts-diamond/
17. Ruparelia, N.B. (2010). Software development lifecycle models. *ACM SIGSOFT Software engineering notes, 35*(3),8 –13.
18. Business Dictionary. (2018). *Organizational culture*. Pridobljeno 5. septembra 2018 iz http://www.businessdictionary.com/definition/organizational-culture.html
19. Dallas, I. & Thandar Wynn, M. (2014). Business Process Management in Small Business: A case study. Pridobljeno 10. oktobra 2017 iz https://www.springerprofessional.de/business-process-management-in-small-business-a-case-study/4099500
20. Desel, J., Oberweis, A. & Van Der Alast, W. (2000). *Business Process Management Models, Techniques and Empirical Studies.* Berlin: Springer.

21. Peterson, S.D., Jaret, P.E. & Findlay Schenck, B. (2016). *Business plans kit for dummies.* New Jersey: John Wiley & Sons, Inc.
22. Dumas, M., La Rosa, M., Mendling, J. & Reijers, H.A. (2013). *Fundamentals of Business Process Management*. Berlin: Springer.
23. Lewis, W.E. (2005). *Software testing and continuous quality improvement*. New York: Auerbach publications.
24. Porter, M.E. (1998). *Competitive advantage creating and sustaining superior preformance.* New York: Free Press.
25. Chang, J.F. (2006). *Business process management systems, Strategy and implementation.* New York: Auerbach Publication.
26. Finances Online reviews for business. (2017). *Business process management software*. Pridobljeno 20. oktobra 2017 iz https://business-process-management.financesonline.com/
27. Finances Online reviews for business. (2018a). *Bizagi REVIEW.* Pridobljeno 12. oktobra 2017 iz https://reviews.financesonline.com/p/bizagi/
28. Finances Online reviews for business. (2018b). *Business process management software*. Pridobljeno 20. oktobra 2017 iz https://business-process-management.financesonline.com/
29. Freund, J., & Rucker, B. (2012). *Real – Life BPMN, Using BPMN 2.0 to Analyze, Improve, And Automate Process in Your Company*. Berlin: Camunda.
30. Gartner. (2018). *Business process management (BPM)*. Pridobljeno 17. marca 2018 iz https://www.gartner.com/it-glossary/business-process-management-bpm
31. Gerth, C. (2013). *Business Process Models, Change Management.* Berlin: Springer.
32. Ghahrai, A. (2018, 25. julij) . Software development life cycle – SDLC Phases. *TESTING EXCELLENCE*. Pridobljeno 2. julija 2018 iz https://www.testingexcellence.com/software-development-life-cycle-sdlc-phases/
33. Gradišar, M., Jaklič, J., & Turk, T. (2007). *Osnove poslovne informatike*. Ljubljana: Ekonomska fakulteta.
34. Grover, V., Jeong Ryul, S., Kettinger, W.J., & T.C.Teng, J. (2015). The Implementation of Business Process Reengineering. *Journal of Management Information Systems, 12*(1), 109 –149.
35. Guru99. (2018). *FUNCTIONAL Testing Tutorial: What is, Process, Types, & Examples*. Pridobljeno 13. julija 2018 iz https://www.guru99.com/functional-testing.html
36. HN Computing. (2018). *Testing Stages*. Pridobljeno 10. junija 2018 iz https://www.sqa.org.uk/e-learning/SDPL03CD/page_18.htm
37. Investopedia. (2018). *Business model*. Pridobljeno 8. aprila 2018 iz https://www.investopedia.com/terms/b/businessmodel.asp
38. ITFAAT. (2018). *CMMI–DEV & SVC*. Pridobljeno 20. marca 2018 iz http://itfaat.com/cmmi_dev/
39. Izbrano podjetje. (2018). *Uradna spletna stran podjetja.* Ljubljana: Izbrano podjetje.
40. Izbrano podjetje. (2015). *Popis postopka za Code review* (interna dokumentacija).

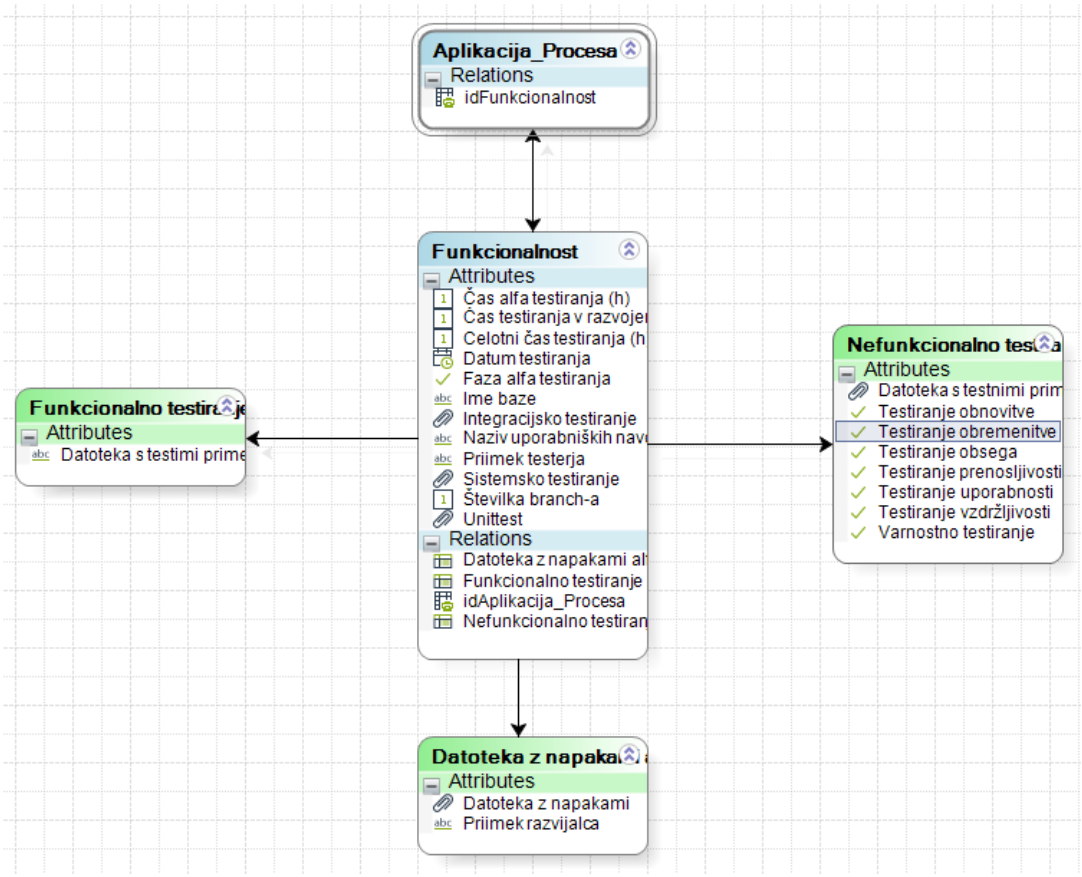
Ljubljana: Izbrano podjetje.
41. Izbrano podjetje. (2016a). *Specifikacija za cikel razvoja programske opreme* (interna dokumentacija). Ljubljana: Izbrano podjetje.
42. Izbrano podjetje. (2016b). *Opis razvojnega Jira Workflow–a* (interna dokumentacija). Ljubljana: Izbrano podjetje.
43. Izbrano podjetje. (2018a). *Tehnične in uporabniške specifikacije različnih projektov* (interna dokumentacija). Ljubljana: Izbrano podjetje.
44. Izbrano podjetje. (2018b). *Organizacijska struktura podjetja* (interna dokumentacija). Ljubljana: Izbrano podjetje.
45. Jeston, J., & Nelis, J. (2014). *Business process management practical guidelines to successful implementations*. New York: Oxon.
46. Kovačič, A. (1998). *Informatizacija poslovanja*. Ljubljana: Ekonomska fakulteta.
47. Kovačič, A., Jaklič, J., Indihar Štemberger, M., & Groznik, A. (2004). *Prenova in informatizacija poslovanja*. Ljubljana: Ekonomska fakulteta.
48. Pratt, M. K. & White, S. K. (2018, 16. julij). What is business analyst? A key role for business – IT efficiency. *CIO FROM IDG*. Pridobljeno 20. julija 2018 iz https://www.cio.com/article/2436638/careers-staffing/project-management-what-do-business-analysts-actually-do-for-software-implementation-projects.html
49. Križevnik, M. & Branko Jurič, M. (2009). Modeliranje in izvajanje poslovnih procesov v storitveno orientiranih arhitekturah. *Uporabna informatika, 3*(3), 137–147.
50. Lusk, S., Paley, S. & Spanyi, A. (2005). The Evolution of Business Process Management as a Professional Discipline. *BPTrends*. Pridobljeno 30. novembra 2017 iz https://www.bptrends.com/publicationfiles/06-05%20WP%20ABPMP%20Activities%20-%20Lusk%20et%20al2.pdf
51. Management.com. (2017). *Leavitt's diamond – A study*. Pridobljeno 15. decembra 2017 iz https://www.tools4management.com/article/leavitts-diamond-a-study/
52. Majumdar, A., Ashique–Ur–Rouf, M., Islam, N. & Arefeen, S. (2011). Capability maturity model integration (CMMI). *International Journal of computer and information technology (IJCIT), 3*(1), 68–74.
53. Moore, C., Finn, K., Khoshafian, S., Winkler, K., Ward–Dutton, N., Kowalowski, F., Swenson, K. & Palmer, N. (2017). *Digital Transformation with Business Process Management*. USA: Future strategies.
54. Meczekalska, M. (2015, 4. September). *Test all the things. Types and examples of software tests. Netguru*. Pridobljeno 7. julija 2018 iz https://www.netguru.co/blog/software-testing
55. Winkler, K. (2018). 6 SIGMA and BPM. *NSI*. Pridobljeno 9. decembra 2018 iz http://www.nsisoluciones.com/eng/index.php/articulos/6-sigma-y-bpm
56. Object management group. (2018). *Business process model & notation (BPMN)*. Pridobljeno 25. avgusta 2017 iz https://www.omg.org/bpmn/
57. O'Sullivan, K. (2008). *Strategic knowledge management in multinational organizations*. New York: New York Institute of technology.
58. Placa.si. (2018). *Plače glede na delovno mesto*. Pridobljeno 6. julija 2018 iz

https://www.placa.si/salaryinfo
59. Rožanc, I. & Mahnič, V. (2003). Uporaba modela CMM v majhnih organizacijah za razvoj programske opreme. *Elektotehniški vestnik, 70*(3), 149–154.
60. Simchi–Levi, D., Kaminsky, P., Simchi–Levi, E. & Shankar, R. (2007). *Designing and managing the supply chain*. California: Future strategies.
61. SMART BEAR. (2018). *What is Unit Testing?* Pridobljeno 15. maja 2018 iz https://smartbear.com/learn/automated-testing/what-is-unit-testing/
62. Software Advice. (2018). *BPM software*. Pridobljeno 22. januarja 2018 iz https://www.softwareadvice.com/business-management/#buyers-guide
63. Stackify. (2018). *What is SDLC? Understand the software development life cycle*. Pridobljeno 10. maja 2018 iz https://stackify.com/what-is-sdlc/
64. Technopedia. (2018). *Software testing*. Pridobljeno 23. junija 2018 iz https://www.techopedia.com/definition/17681/software-testing
65. The Economic Times. (2018). *Definition of 'Software testing'*. Pridobljeno 25. junija 2018 iz https://economictimes.indiatimes.com/definition/software-testing
66. TRY QA. (2018). *What are software testing levels?* Pridobljeno 13. maja 2018 iz http://tryqa.com/what-are-software-testing-levels/
67. Vukšić–Bosilj, V., Hernaus, T. & Kovačič, A. (2008). *Upravljanje poslovnim procesima, organizacijski i informacijski pristup.* Zagreb: Školska knjiga.
68. Weske, M. (2007). *Business Process Management, Concepts, Languages, Architectures.* Berlin: Springer.
69. Workflow Management Coalition. (2018). *What is BPM?* Pridobljeno 15. marca 2018 iz https://www.wfmc.org/what-is-bpm
70. Xiao, J. (2016. oktober). Life cycle of business process management. *Big think.* Pridobljeno 10. oktobra 2017 iz http://bigthink.com/articles/life-cycle-of-business-process-management

PRILOGE

Priloga 1: Podatkovni model za izgradnjo aplikacije

Slika 1: Podatkovni model aplikacije v Bizagi Studiu



Vir: lastno delo.

Priloga 2: Podrobnejši rezultati časovne analize obstoječega procesa testiranja programskih rešitev za štiri iteracije v programu Bizagi Modeler

Tabela 1: Podrobnejši rezultati časovne analize obstoječega procesa testiranja programskih rešitev

Naziv aktivnosti	Število izvedenih iteracij	Min. čas izvedbe aktivnosti	Max. čas izvedbe aktivnosti	Povprečni čas izvedbe aktivnosti	Celotni čas izvedbe aktivnosti
Obstoječi proces testiranja programskih rešitev	**4**	**8h 34min 55s**	**1dan 17h 54min 51s**	**19h 54min 57s**	**3dni 5h 11min 19s**
Prejem obvestila	4	1min 39s	4min 39s	3min 26s	13min 45s
Pregled tehnične dokumentacije	1	2h 15min 37s	5h 1min 55s	3h 52min 24s	3h 29min 38s
Izvedba verifikacije programske rešitve	4	1h 23min 28s	2h 10min 18s	1h 47min 8s	7h 8min 34s
Izvedba validacije programske rešitve	4	1h 30min 23s	2h 12min 47s	1h 46min 50s	7h 7min 21s
Izvedba pregleda kode	4	3h 24min	4h 19min 28s	3h 56min 2s	15h 44min 8s
Sporočilo, da je koda funkcionalnosti primerna za osnovni test	3	20min 37s	26min 12s	23min 36s	1h 10min 48s
Izvedba osnovnega testa funkcionalnosti	3	3h 19min 38s	3h 31min 43s	3h 27min 15s	10h 21min 45s
Sporočilo, da je funkcionalnost primerna za izvedbo alfa testiranja	1	13min 30s	13min 30s	13min 30s	13min 30s
Prejem obvestila	3	2min 28s	4min 37s	3min 15s	9min 45s
Beleženje napak	1	1h 44m 27s	1h 44min 27s	1h 44min 27s	1h 44min 27s
Izvedba ročnega testiranja funkcionalnosti	4	14h 1min 17s	14h 1min 17s	14h 1min 17s	14h 1min 17s
Priprava testnih scenarijev s pričakovanimi rezultati	3	4h 6min 42s	4h 6min 42s	4h 6min 42s	4h 6min 42s
Priprava ustreznega testnega okolja	1	57min 37s	57min 37s	57min 37s	57min 37s
Pregled tehnične in uporabniške specifikacije	1	5h 2min 10s	5h 2min 10s	5h 2min 10s	5h 2min 10s
Prejem obvestila	4	3min 30s	3min 30s	3min 30s	14min
Dopolnjevanje uporabniške specifikacije	1	3h 34min 39s	3h 34min 39s	3h 34min 39s	3h 34min 39s
Pošiljanje obvestila	1	19min 30s	19min 30s	19min 30s	19min 30s
Prejem obvestila	1	3min 23s	3min 23s	3min 23s	3min 23s
Pošiljanje nabora napak	3	21 min 20s	18min 30s	18min 20s	55min
Prejem obvestila	3	3min 32s	3min 32s	3min 32s	10min 36s
Sporočilo, da je koda funkcionalnosti pripravljena za pregled	4	4min 24s	5min 27s	4min 46s	19min 5s

Vir: lastno delo.

Priloga 3: Podrobnejši rezultati časovne analize prenovljenega procesa testiranja programskih rešitev za dve iteraciji v programu Bizagi Modeler

Tabela 2: Podrobnejši rezultati časovne analize prenovljenega procesa testiranja programskih rešitev

Naziv aktivnosti	Število izvedenih iteracij	Min. čas izvedbe aktivnosti	Max. čas izvedbe aktivnosti	Povprečni čas izvedbe aktivnosti	Celotni čas izvedbe aktivnosti
Prenovljeni proces testiranja programskih rešitev	**2**	**17h 54min 17s**	**1 dan 9h 28min 31s**	**1 dan 1h 41min 24s**	**2 dni 9h 54min 20 s**
Dokumentiranje napak	2	16min	17min 38s	16min 49s	33min 38s
Zagon testnih primerov	2	26min 11s	27min 31s	26min 51s	53min 43s
Dokumentiranje napak integracijskega testiranja	2	15min 8s	15min 54s	15min 31s	31min 2s
Izvedba funkcionalnega testiranja	1	6h 5min 11s	6h 5min 11s	6h 5min 11s	6h 5min 11s
Izbira nefunkcionalnega testiranja	1	11min 39s	11min 39s	11min 39s	11min 39s
Prejem obvestila	1	2min 36s	2min 36s	2min 36s	2min 36s
Beleženje napak funkcionalnega testiranja	1	1h 43min 5s	1h 43min 5s	1h 43min 5s	1h 43min 5s
Pregled tehnične in uporabniške specifikacije	1	3h 26min 5s	3h 26min 5s	3h 26min 5s	3h 26min 5s
Priprava ustreznega testnega okolja	1	47min 53s	47min 53s	47min 53s	47min 53s
Priprava testnih scenarijev s pričakovanimi rezultati	1	3h 24min 56s	3h 24min 56s	3h 24min 56s	3h 24min 56s
Izvedba nefunkcionalnega testiranja	1	4h 43min 21s	4h 43min 21s	4h 43min 21s	4h 43min 21s
Beleženje rezultatov nefunkcionalnega testiranja	1	59min	59min	59min	59min
Zagon testiranja komponent	2	15min 45s	25min 13s	20min 29s	40min 58s
Priprava testnih primerov za integracijsko testiranje	2	2h 30min 27s	3h 54min 55s	3h 12min 41s	6h 25min 23s
Obveščanje o primernosti kode za izvedbo testiranja	2	2min 25s	2min 41s	2min 33s	5min 7s
Izvedba validacije	2	1h 14min 26s	1h 32min 9s	1h 23min 18s	2h 46min 36s
Priprava testnih scenarijev za testiranje komponent	2	2h 53min 16s	4h 28min	3h 40min 38s	4h 21min 7s
Izvedba pregleda kode	2	1h 22min 56s	2h 8min 1s	1h 45min 28s	3h 30min 57s
Izvedba verifikacije	2	1h 22min 56s	2h 8min 1s	1h 45min 28s	6min 35s
Obveščanje, da je koda funkcionalnosti poslana v pregled	2	1min 39s	4min 56s	3min 17s	3h 30min 57s
Prejem obvestila	2	2min 6s	2min 13s	2min 10s	4min 20s
Pregled tehnične specifikacije za funkcionalnost	2	2h 42min 50s	2h 51min 45s	2h 47min 17s	5h 34min 35s
Prejem obvestila	2	46s	1min 12s	59s	1min 58s
Izbira vrste nefunkcionalnega testiranja	1	3min 27s	3min 27s	3min 27s	3min 27s
Priprava testnih scenarijev	1	2h 27min 2s	2h 27min 2s	2h 27min 2s	2h 27min 2s
Izvedba sistemskega testiranja	1	1h 33min 38s	1h 33min 38s	1h 33min 38s	1h 33min 38s
Dokumentiranje izidov sistemskega testiranja	2	23min 8s	27min 41s	25min 24s	50min 49s
Pošiljanje obvestila	1	1min 18s	1min 18s	1min 18s	1min 18s
Priprava uporabniških navodil	1	3h 30min 10s	3h 30min 10s	3h 30min 10s	3h 30min 10s
Pošiljanje obvestila o naboru napak	1	2min 30s	2min 30s	2min 30s	2min 30s
Naložitev datotek v aplikacijo	1	8min 20s	8min 20s	8min 20s	8min 20s
Naložitev datotek v aplikacijo	2	6min 6s	7min 42s	6min 54s	13min 48s

Vir: lastno delo.