UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

MAGISTRSKO DELO

ANALIZA INFORMACIJSKE REŠITVE ZA E-POSLOVANJE IZBRANEGA ZAVODA

Ljubljana, 7. februar 2019

MITJA DROFENIK

IZJAVA O AVTORSTVU

Podpisani Mitja Drofenik, študent Ekonomske fakultete Univerze v Ljubljani, avtor predloženega dela z naslovom Analiza informacijske rešitve za e-poslovanje izbranega zavoda, pripravljenega v sodelovanju s svetovalcem prof. dr. Mirkom Gradišarjem,

I Z J A V L J A M

1. da sem predloženo delo pripravil samostojno;

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

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

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

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

6. da sem pridobil vsa potrebna dovoljenja za uporabo podatkov in avtorskih del v predloženem delu in jih v njem jasno označil/-a;

7. da sem pri pripravi predloženega dela ravnal/-a v skladu z etičnimi načeli in, kjer je to potrebno, za raziskavo pridobil soglasje etične komisije;

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

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

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

V Ljubljani, dne _____ Podpis študenta(-ke): _____

KAZALO

KAZALO TABEL

KAZALO SLIK

SEZNAM KRATIC

ang. – angleško

RIP – (ang. Routing Information Protocol); računalniška izmenjava podatkov

FTP – (ang. File Transfer Protocol); protokol za prenos datotek

WWW – (ang. World Wide Web); svetovni splet

NFC – (ang. Near Field Communication); komunikacijska tehnologija kratkega dosega

HTML – (ang. Hypertext Markup Language); jezik za označevanje nadbesedila

CSS – (ang. Cascading Style Sheets); kaskadne oblikovne predloge

AJAX – (ang. Asynchronous JavaScript and XML); asinhroni JavaScript in XML

SPA – (ang. Single Page Application); enostranska aplikacija

SASS – (ang. Syntactically Awesome Style Sheets); sintaktične oblikovne predloge

DOM – (ang. Document Object Model); model predmeta dokumenta

TAD – (ang. Tabular Appliactaion Development); tabelarni razvoj aplikacij

RUP – (ang. Rational Unified Process); racionalni združeni proces

FDD – (ang. Feature Driven Development); funkcijsko osredotočen razvoj

DSDM – (ang. Dynamic System Developent Method); metoda razvoja dinamičnega sistema

ASD – (ang. Adaptive Software Development); adaptivni razvoj programske opreme

XP – (ang. Extreme Programming); ekstremno programiranje

UML – (ang. Unified Modeling Language); enoten modelski jezik

CBA – (ang. Cost Benefit Analysis); analiza stroškov in koristi

ROI – (ang. Return on Investment); donosnost naložbe

UVOD

Charles Darwin je že v 19. stoletju predstavil temelje teorije, ki pravi, da so se vsa živa bitja zaradi mehanizma naravne selekcije primorana neprestano razvijati in spreminjati, preživijo pa le tista, ki se najbolj učinkovito prilagodijo spremembam v okolju. Podobno dogajanje se neprestano odvija tudi v poslovnem svetu, kjer poslovni subjekti s svojimi poslovnimi odločitvami neprestano iščejo konkurenčno prednost pred svojimi tekmeci. Za konkurenčne poslovne odločitve pa so potrebne ustrezne in pravočasne informacije.

V enem letu se na celem svetu ustvari več kot dva milijona terabajtov edinstvenih informacij, številke pa iz leta v leto še naraščajo (Chung, Chen & Jr, 2005). Poslovni subjekti morajo prepoznati zanje pomembne informacije ter jih uporabiti kot prednost pred konkurenco na trgu.

V poslovnem svetu igra ključno vlogo raba omejenih resursov, med katerimi je eden najpomembnejših čas. V svojem delu ga omenja tudi Kovačič A. (1998, str. 12), ki ugotavlja, da se čas močno razlikuje od tradicionalnih proizvodnih dejavnikov, saj ga ne moremo shraniti, če je le-tega preveč. Čas, ki ga produktivno ali neproduktivno porabimo, je za vedno izgubljen. Vsi poznamo tudi pregovor »čas je denar«, ki v današnjem poslovnem svetu še kako drži, saj si vsak poslovni subjekt prizadeva čim bolj optimizirati poslovne procese in s tem prihraniti čim več časa oziroma ga nameniti za druge aktivnosti.

Kovačič (1998, str. 40) navaja, da raziskave na področju zagotavljanja konkurenčne prednosti podjetja z ustrezno razvito informacijsko tehnologijo kažejo, da ta predstavlja eno redkih poslovnih priložnosti, ki jih ima podjetje na voljo v boju s svojo konkurenco na tržišču. Enako v svojem delu navaja tudi Cerovšek (2012).

V zadnjih letih na življenje posameznika in tudi na podjetja vplivajo hitre spremembe, ki se dogajajo v našem okolju. Z razvojem informacijske tehnologije se odpirajo nove priložnosti, hkrati pa ta podjetja sili, da svoje poslovanje prilagajajo novim razmeram in uporabi novih digitalnih orodij.

Hitre spremembe v okolju pa tako za velika kot tudi majhna in srednja podjetja predstavljajo velik napor, saj nimajo primernega znanja in virov, da bi jim sledila. Tudi odpor do sprememb je znotraj podjetij navadno velik, saj si zaposleni ne želijo uvajanja novih poslovnih procesov in delovnih nalog.

Gradišar, Jaklič in Turk (2012, str. 12) v svojem delu navajajo vsaj tri razloge, zaradi katerih mora poslovanje temeljiti na informacijski tehnologiji:

- Informacija je ključni vir konkurenčne prednosti in zaradi informacijske tehnologije ta prihaja na prvo mesto ter poslovnemu subjektu omogoča najhitrejšo rast in najbolj dinamične spremembe.

- Informacijska tehnologija je ključna pri učinkovitem izvajanju dejavnosti neke organizacije in je temelj boljšega odločanja:

    - krajša čas, ki ga zaposleni porabijo za izvedbo dane aktivnosti,
    - zmanjšuje potrebo po zalogah, opremi, denarju in ljudeh,
    - izboljšuje delo s strankami na eni in dobavitelji na drugi strani ter omogoča hitrejše sledenje spremembam na trgu,
    - poenostavlja pretok informacij med organizacijskimi ravnmi,
    - povečuje znanje organizacije ter ustvarja pogoje za učenje in delitev znanja.
- Internetna doba organizacijam omogoča prenos dela poslovanja ali celotnega poslovanja na spletne strani.

E-poslovanje ni nič drugega kot avtomatizacija poslovnih procesov, ti pa so pri različnih avtorjih različno opredeljeni:

- Champy in Hammer (1993, str. 45) v svojem delu navajata, da je poslovni proces seštevek dejavnosti, ki zahtevajo eno ali več vrst vložkov in ustvarjajo rezultate, ki prinašajo neko vrednost.
- Khan (204, str. 67) navaja, da je poslovni proces skupek zaporednih in vzporednih nalog, ki jih ljudje in aplikacije izvajajo z namenom doseganja nekega skupnega cilja.
- Bosilj in Kovačič (2005, str. 29) pa sta zapisala, da je poslovni proces opredeljen kot logična povezava med seboj povezanih izvajalskih in nadzornih aktivnosti, katerih posledica je proizvod (izdelek ali storitev). Navadno poslovni procesi ne potekajo v eni organizacijski enoti, pač pa posamezne aktivnosti izvajajo različni oddelki.

Z avtomatizacijo poslovnih procesov se ti prenovijo in čim bolj poenostavijo z namenom, da bi delovni tok potekal kar se da nemoteno. Pri tem se sprostijo tudi dragoceni človeški resursi, ki se zaradi optimizacije delovnega procesa lahko posvetijo drugim opravilom, s tem pa tudi poslovni subjekt postane bolj prilagodljiv in konkurenčen (Gradišar, Jaklič, Damij & Baloh, 2005).

V izbranem zavodu nimajo enotnega delovnega toka izdaje in potrjevanja naročilnic. Odvisen je od vsake organizacijske enote posebej, kar pa povzroča zmedo znotraj zavoda. Ker pa je treba ob uvedbi elektronskega poslovanja proces poenotiti, saj mora biti enoličen in znan vsem, je potrebna detajlna analiza celotnega procesa in informacijske rešitve.

Cilj magistrskega dela je analiza uvedbe naročilnic v elektronsko poslovanje izbranega zavoda. Zaradi poenotenja poslovnega procesa sem analiziral primarne podatke in delovni tok po različnih organizacijskih enotah, določil njihove skupne točke in poiskal posebnosti ter predlagal nov delovni tok, ki bo vsem enak.

Namen je povečati stopnjo elektronskega poslovanja v izbranem zavodu z vpeljavo informacijske rešitve za razbremenitev zaposlenih v administraciji. S tem se bo povečala tudi hitrost poslovanja, hkrati pa se bodo zmanjšali stroški.

Izdelava magistrske naloge temelji na domači in tuji literaturi s področja elektronskega poslovanja in njegovega razvoja. Velik del literature je pridobljen iz svetovnega spleta zaradi teme magistrskega dela. Analiza informacijske rešitve sloni na znanju, ki sem ga pridobil med študijem na Ekonomski fakulteti in izkušnjami, ki sem jih osvojil pri delu na področju elektronskega poslovanja v Podjetju informacijskega inženiringa z izvajanjem vsakodnevnih nalog na delovnem mestu. Temeljna metoda dela je analiza obstoječega sistema naročanja materiala preko naročilnic za lažji predlog informacijske rešitve.

V prvem delu je opisan razvoj elektronskega poslovanja, kaj zavod z uvedbo elektronskega poslovanja pridobi, vpliv na poslovanje in procese ter priložnosti za nadaljnji razvoj in izboljšave. V drugem in tretjem delu je temeljito analiziran delovni tok naročanja materiala preko naročilnic v izbranem zavodu, predlagan pa je tudi nov delovni tok, ki najbolj ustreza njihovi organizacijski strukturi in izkorišča tehnološke možnosti.

1 ELEKTRONSKO POSLOVANJE

1.1 Pojem elektronsko poslovanje

Elektronsko poslovanje (v nadaljevanju e-poslovanje) ne pomeni nič drugega kot poslovati elektronsko oziroma natančneje poslovati s pomočjo informacijske in komunikacijske tehnologije. Glede na stopnjo digitalizacije tako poslovanje delimo na različne stopnje. Organizacije lahko poslujejo povsem klasično ali povsem elektronsko. Največ je takih, ki so nekje vmes in oboje kombinirajo, saj postopoma uvajajo digitalizacijo (Razgoršek & Potočar, 2009).

Za definicijo e-poslovanja obstaja širok spekter razlag. Pojem se v angleščini (»electronic commerce«) navezuje bolj na elektronsko prodajo, slovenska oznaka e-poslovanja pa ta pomen vsekakor presega, saj vključuje še vrsto drugih oblik uporabe. V e-poslovanje so vključene vse oblike informacijske in komunikacijske tehnologije poslovnih odnosov. Pojavlja se v trgovinskih, proizvodnih in storitvenih organizacijah ter pri potrošnikih, ponudnikih informacij in v državni upravi (Jerman-Blažič, Klobučar, Perše & Nedeljković, 2001, str. 11).

Vsi zgoraj našteti poslovni odnosi pa vsebujejo spodnje elemente (Jerman-Blažič, Klobučar, Perše & Nedeljković, 2001, str. 11):

- način dela (računalniška izmenjava podatkov ob uporabi omrežja, kot je internet),
- vsebina poslovanja (prodaja blaga in storitev ali informacij, plačevanje, bančne transakcije, izmenjava dokumentov, storitve trženja in komuniciranja, podpora poslovnemu informacijskemu sistemu organizacij, nakupovanje na spletu, opravljanje dela in nudenje pomoči na daljavo, izvajanje pouka na daljavo ter storitve državne uprave in podobno),

- udeleženci poslovanja (podjetniki, raziskovalci, menedžerji, občani, delavci, študenti, dijaki, učitelji, podjetja, bolnišnice, muzeji, galerije, univerze, izobraževalne in državne ustanove).

Ravi Kalakota v svojem delu opredeljuje e-poslovanje kot izvajanje poslovnih transakcij s pomočjo računalniško podprte izmenjave podatkov preko omrežij. Sodobnejše razlage definirajo e-poslovanje kot proces kupovanja, prodaje in izmenjave vseh različnih proizvodov, storitev in informacij preko interneta kot omrežnega okolja. Najbolj preprosto lahko e-poslovanje definiramo kot »poslovati elektronsko«. Razlage e-poslovanja lahko zaradi obsežnosti pojma v slovenskem jeziku preučujemo z različnih aspektov (Kalakota & Whinston, 1997, str. 97):

- komunikacijski pogled: prenos vseh informacij ter storitev preko telefonske linije, računalniškega omrežja ali drugih komunikacijskih povezav;
- poslovno-procesni pogled: v poslovni proces vnaša avtomatizem s pomočjo aplikacij;
- storitveni pogled: poveča se učinkovitost poslovanja, prav tako pa tudi kvaliteta in izvedba storitve oz. dobava, zmanjšajo pa se stroški;
- povezovalni pogled: deluje v omrežjih, ki so med seboj povezana brez posrednikov.

Zakaj elektronsko poslovanje? Danes to ne sme več biti vprašanje, ampak stvarnost, s katero se srečujemo tako rekoč na vsakem koraku. Torej danes ne obstaja več vprašanje zakaj, ampak kako elektronsko poslovati, da bomo lahko čim bolje izkoristili prednosti, ki nam jih ponuja. Podjetja morajo najti način, kako elektronsko poslovanje najbolj uspešno vpeljati v poslovanje podjetja (Razgoršek & Potočar, 2009).

Zato se v poslovanje podjetja vsako leto bolj vpleta tudi e-poslovanje. Gre za poslovanje, ki zajema vse oblike komunikacijske in informacijske tehnologije v poslovnih odnosih. Na tak način se tudi izboljša komunikacija s potrošniki ter optimizira tržna pot, kar pa je lahko že odločilna prednost pred konkurenco (Nahtigal, 2010).

Za uspešno implementacijo e-poslovanja je potrebno poznavanje najnovejših trendov v razvoju informacijske tehnologije, temeljito poznavanje trenutnega delovanja, upoštevanje temeljnih pravil v načrtovanju ter druga znanja s področij marketinga, financ, ekonomije ipd. Potrebna so nova znanja ter premik iz okvirov vsakdanjosti, brez katerih si celovite implementacije e-poslovanja ne moremo zamisliti (Nahtigal, 2010).

Elektronsko poslovanje se pojavlja v bančništvu, zavarovalništvu, trženju, trgovanju in spletni trgovini, računalniško podprtem skupinskem delu in pouku na daljavo.

Vse zgoraj naštete dejavnosti pa vsebujejo spodnje elemente (Jerman-Blažič, Klobučar, Perše & Nedeljković, 2001):

- način dela (računalniška izmenjava podatkov ob uporabi omrežja, kot je internet),

- vsebina poslovanja (prodaja blaga in storitev ali informacij, plačevanje, bančne transakcije, izmenjava dokumentov, storitve trženja in komuniciranja, podpora poslovnemu informacijskemu sistemu organizacij, nakupovanje na spletu, opravljanje dela in nudenje pomoči na daljavo, izvajanje pouka na daljavo ter storitve državne uprave in podobno),
- udeleženci poslovanja (podjetniki, raziskovalci, menedžerji, občani, delavci, študenti, dijaki, učitelji, podjetja, bolnišnice, muzeji, galerije, univerze ter druge izobraževalne in državne ustanove).

Pri vsem tem pa se v elektronskem poslovanju uporablja mednarodno dogovorjen standardni format, kar omogoča komunikacijo med katerim koli kupcem in dobaviteljem na svetu.

1.2 Razvoj elektronskega poslovanja

Elektronsko poslovanje se je prvič pojavilo konec 70. let prejšnjega stoletja. Takrat se je začel razvoj računalniških omrežij in interneta ter uvajanje standardov za računalniško izmenjavo podatkov RIP (ang. Routing Information Protocol). Takrat še ni bilo slutiti, s kolikšno hitrostjo in kako intenzivno bo razvoj informacijske tehnologije vplival na spremembo načina življenja in poslovanja. V tem času je RIP zahtevala veliko znanja in finančnih sredstev, zato ni bila dostopna srednjim in malim podjetjem ter posameznikom. (Turban, McLean & Wetherbe, 1999).

V prvih elektronskih prenosih, ki so spremenili načine poslovanja na finančnih trgih, so sodelovale banke, ki so za transakcije uporabljale varna zasebna omrežja. Zaradi elektronskih prenosov je veliko podatkov iz papirnate oblike prešlo v elektronsko obliko. V zgodnjih 80. letih se je e-poslovanje v podjetja razširilo v obliki sistemov za prenos datotek in elektronske pošte. RIP je povezovala izbrana podjetja, ki so preko zasebnih omrežij izmenjevala podatke, kot so naročila, plačilni nalogi, računi in podobno (Jerman-Blažič, Klobučar, Perše & Nedeljković, 2001, str. 13–14).

Takšna praksa elektronske izmenjava podatkov je pokazala nekaj slabosti (Škrlec, 2002, str. 18):

- Širšemu krogu je bila težje dostopna.
- Specifične komunikacijske povezave med podjetji, ki so bile nedostopne majhnim podjetjem.
- Zaradi natančno določene oblike podatkov in dokumentov, ki se izmenjujejo na podlagi natančno določenih standardov, je bila preslikava v pravilno obliko draga in zapletena.

Kljub zgoraj naštetim pomanjkljivostim takšnega sistema je bila dobra stran tovrstne komunikacijske infrastrukture in standardizacije ta, da se je razvila avtomatizacija procesov, ki je bistveno skrajšala čas obdelave podatkov in tudi zmanjšala stroške prenosa.

Vse to se je dogradilo v nadaljnjem razvoju, ki je bil hiter in učinkovit. Nove rešitve so se tako dopolnjevale s staro tehnologijo, kar je podjetjem omogočilo cenovno ugodno nadgradnjo njihovega sistema. E-poslovanje se je začelo intenzivneje razvijati šele v 90. letih z razmahom interneta in javnega omrežja, ki omogoča vključevanje večjega števila uporabnikov. Tako je internet omogočil vključevanje in medsebojno povezovanje vseh potrošnikov, podjetij in države oziroma državne uprave. Glede na trenutni obseg e-poslovanja v družbi lahko govorimo o elektronski družbi (e-družbi) oziroma kar o informacijski družbi (Nahtigal, 2010).

Temelji preproste uporabe izhajajo iz ključnih predpostavk za gradnjo interneta, ki so sledeče (Škrlec, 2002):

- Omrežje, ki je dosegljivo kjer koli in kadar koli.
- Povezovanje velikega števila navzven odprtih manjših računalniških omrežij, ki nimajo skupnega lastnika.
- Preprosta uporaba z že vnaprej določenimi postopki.
- Strojna in programska neodvisnost.

Internet je tako znižal stroške in povečal učinkovitost komuniciranja, odprl pot do novega načina poslovanja, združil trge, skrajšal čas postopkov, omogočil lažje prilagajanje spremembam na trgu, omogočil tudi vpeljavo večpredstavnih vsebin ter zagotovil, da so podjetja postala globalno povezana (Ošlak, 2005).

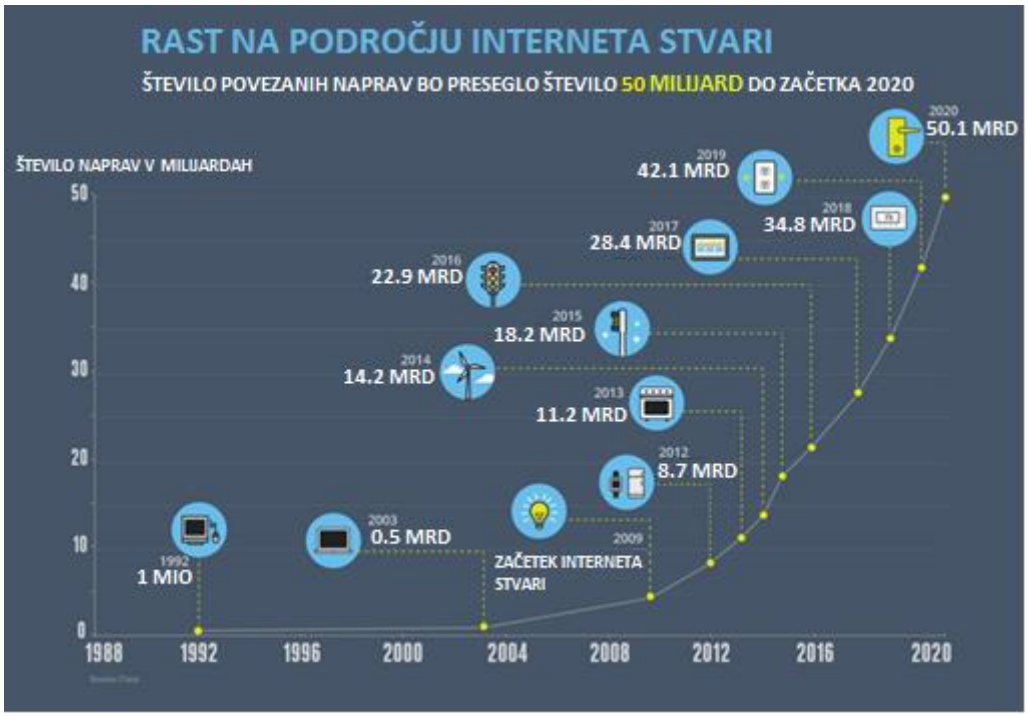
Svetovni splet tako malim podjetjem omogoča, da lahko na bolj enaki tehnološki podlagi kot pred desetletji tekmujejo z velikimi mednarodnimi podjetji. Vsako majhno podjetje tako preko interneta z minimalnimi investicijami vstopi v infrastrukturo na večmilijonskem trgu. Za osnovo so potrebni osebni računalnik, modem in povezava z internetom. Novi standardi pa izpodrivajo tudi osebni računalnik, saj so pametni telefoni in tablični računalniki že dovolj zmogljivi za vstop na trg.

Tako svetovni splet omogoča nastajanje novih vrst podjetij, ki jih imenujemo virtualna podjetja oziroma navidezna podjetja. Taka podjetja nastanejo izključno na internetu, saj ni potrebe po fizični interakciji s kupci in dobavitelji (Razgoršek & Potočar, 2009).

Največji razvoj v elektronskem poslovanju se je začel po letu 1996, ko je tehnologija interneta dozorela in postala javno dostopna (Jerman-Blažič, Klobučar, Perše & Nedeljković, 2001).

Slika 1 prikazuje rast uporabe naprav, povezanih z internetom, kjer je dobro razvidno, da se je rast začela šele po letu 2000. V začetnih letih so bili to večinoma le osebni računalniki, kasneje pa se je internet naselil v večino naprav, ki jih uporabljamo sedaj, od pametnih križišč do pametnih gospodinjskih aparatov (Krum, 2016).

Slika 1: Rast uporabe elektronskega poslovanja



Vir: Krum (2016).

Digitalna tehnologija je takrat prvič omogočila, da se lahko slika, zvok, video in znaki kombinirajo, shranjujejo in prenašajo hitro in učinkovito ter v velikih količinah brez tveganja za zmanjšanje kakovosti. Digitalna tehnologija je tako združila tudi nekdaj oddaljene veje gospodarstva, ki so sedaj primorane sodelovati in konkurirati (Razgoršek & Potočar, 2009).

1.3 Klasifikacija elektronskega poslovanja

Kot sem že omenil, je e-poslovanje globalni pojem. Uporabniki imajo tako dostop do informacij, ki so zbrane na drugih delovnih postajah. Računalniško omrežje je sestavljeno iz omrežij računalnikov in strežnikov, skupek le-teh pa predstavlja infrastrukturo za e-poslovanje. Računalniško omrežje delimo na (Gradišar, 2017):

- internet – javno omrežje,
- intranet – zaprto omrežje v okviru ene organizacije oziroma nekaj povezanih organizacij,
- ekstranet – povezuje različne intranete organizacij preko interneta,
- privatna omrežja.

Razvoj e-poslovanja sta pospešila internet in nanj vezane aplikacije. E-poslovanje glede na področje uporabe delimo na (Pucihar, 2018, str. 11–12):

- poslovanje med podjetji (B2B),
- poslovanje med podjetji in potrošniki (B2C),
- poslovanje med potrošniki in podjetji (C2B),
- poslovanje med potrošniki (C2C),
- poslovanje med državo in podjetji (G2B),
- poslovanje med vladnimi organizacijami in državljani (G2C),
- poslovanje med podjetjem in zaposlenim (B2E),
- poslovanje med vladnimi organizacijami (G2G),elektronska prodaja na drobno (e-tailing),
- elektronsko učenje,
- poslovanje preko mobilnega telefona (m-commerce),
- sodelovanje v poslovanju (c-commerce).

Ker uvedba e-poslovanja vpliva na poslovne transakcije tako v oskrbovalni verigi kot tudi s spremembami v marketingu, vse to pripelje do sprememb v celostni strategiji podjetja. Podjetje v elektronskem okolju lahko komunicira z zgoraj naštetimi subjekti na različne načine. Najbolj znani načini so (Turban, 2009, str. 7-8):

- svetovni splet – WWW (ang. World Wide Web),
- elektronska pošta,
- prenos datotek – FTP (ang. File Transfer Protocol),
- novice,
- videokonferenca,
- priključitev na oddaljen računalnik.

Preko omrežja se z uporabo zgoraj navedenih načinov e-poslovanja izbrišejo nacionalne meje in časovne omejitve. Podjetja se tako lahko čez noč spremenijo iz lokalnih v globalna podjetja. Omrežni trg pozna dobra in slaba podjetja ter mu za njihovo velikost ni mar. Kako dobro je podjetje, pa je odvisno od izkoriščenih konkurenčnih prednosti, ki jih e-poslovanje ponuja. Pomembno se je izogibati tudi morebitnim nevarnostim. Z elektronskim poslovanjem se je rek »velike ribe jedo majhne« preoblikoval v »hitre ribe jedo počasne« (Ošlak, 2005).

1.4 Tehnologije za razvoj aplikacij

Z izumom svetovnega spleta, katerega namen je bil izmenjava informacij s pomočjo različnih povezav, se je začel tudi razvoj spletnih aplikacij (Berners-Lee, 1998).

Temelj spleta je označevalni jezik HTML (ang. Hypertext Markup Language). Z njim se je začelo obdobje izdelave statičnih spletnih strani, ki so namenjene podajanju informacij. Zaradi potrebe po oblikovanju spletnih strani je kmalu zatem sledil razvoj kaskadnih oblikovnih predlog oziroma CSS (ang. Cascading Style Sheets) (Bos, 2016), kasneje pa še

programskega jezika JavaScript, ki je pomemben za interakcijo z uporabnikom spletne strani (Peyrott, 2017).

Razvoj spleta je šel v smer izdelave dinamičnih spletnih strani, kot jih poznamo danes. Nastali so različni programski in skriptni jeziki, glavni namen vsega pa je bil, da se uporabniki pozitivno odzovejo na podane informacije. Razlika med dinamično in statično spletno stranjo je, da so statične pripravljene na takojšnji izris, medtem ko se dinamične generirajo sproti. Problem pri sprotnem generiranju je obremenitev strežnika, ker je treba z vsako novo zahtevo celotno prikazano sliko ponovno generirati (Sušnik, 2017).

Delno rešitev za težavo vsakokratnega generiranja so predstavljale strani s spreminjajočimi se deli. Te so tako statične kot dinamične in namesto ponovnega izrisa uporabljajo tehnologije, kot je recimo asinhroni JavaScript in XML oziroms AJAX (ang. Asynchronous JavaScript and XML). S to tehnologijo je omogočeno spreminjanje izključno tistih delov strani, ki jih zahteva uporabnik. Rezultat uporabe tovrstnih strani je zmanjšana obremenitev strežnikov in s tem boljša uporabniška izkušnja, saj uporabnikom ni treba čakati na ponoven izris celotne spletne strani (Sušnik, 2017).

Kljub zgoraj zapisanemu pa so se zaradi težnje po še boljši uporabniški izkušnji pojavile enostranske spletne aplikacije oziroma SPA (ang. Single Page Application), ki imajo v primerjavi s predhodnimi vrstami strani spremenjen način delovanja. Novost je v tem, da predhodnice te vrste strani celotno logiko hranijo na strežniku, medtem ko enostranske aplikacije večino logike prenašajo v sklop odjemalca. Delovanje aplikacije tako ni povsem odvisno od spletnega strežnika, kajti ta je namenjen le zagotavljanju datotek in podatkov za izvajanje aplikacije, ki jih uporabnik potrebuje. Jedro aplikacije pa predstavlja JavaScript, saj skrbi za izvajanje aplikacijske logike (Sušnik, 2017). Enako v svojem članku pojasnjuje tudi Diehl (2013), ki navaja, da se v uporabniško izkušnjo na spletu dodaja vse več JavaSripta, kar je pozitivno, saj pomeni, da se s tem izboljšuje uporabniška izkušnja.

Zaradi prednosti, ki jih prinašajo enostranske spletne aplikacije, in nadomeščanja odsotnosti začetne strukturiranosti strani so se začele pojavljati kombinirane spletne aplikacije. Ob začetnem nalaganju se naloži celotna struktura, ki je prikazana uporabniku, v nadaljnjih akcijah pa se aplikacija obnaša kot enostranska (Sušnik, 2017).

Poznamo še hibridne in napredne aplikacije. Pri hibridnih aplikacijah je namen prodreti na preostale platforme, njihov razvoj pa je enak kot pri SPA. V okviru hibridnih aplikacij se osredotočamo predvsem na mobilne aplikacije. Za izvajanje tovrstnih aplikacij je na mobilnih napravah zadolžen gradnik za prikazovanje spletne vsebine. Napredne aplikacije se od hibridnih razlikujejo predvsem po zmožnostih uporabe funkcionalnosti naprave, omejuje pa jih razširjenost na več platformah. Obe vrsti pa delujeta na osnovi gradnika za prikazovanje spletne vsebine (Sušnik, 2017).

1.4.1 HTML, CSS, SASS

Jezik HTML se uporablja za zapis strukture spletnih strani. Sestoji iz strukturnih elementov, imenovanih značke. HTML je zapisan v drevesni strukturi, značke skupnih predelov strani pa so gnezdene. Brskalnik opravi izris, sledi pa še nalaganje dodatnih sredstev, ki omogočijo, da se stran prikaže v celoti. Najnovejša različica je HTML 5, ki omogoča integracijo spleta z domačim okoljem platforme, kjer je spletna aplikacija postavljena.

CSS se uporablja za definiranje strukturnih elementov s pomočjo sintaksnih pravil. Z njim je določen način izbiranja elementov, ki nato na strani odjemalca omogočajo njihovo oblikovanje. Z njimi lahko nadzorujemo pisavo, ozadja, barve, obliko in postavitev teksta. Slogovne predloge CSS delujejo neodvisno od jezika HTML, saj jih lahko uporabljamo tudi skupaj z drugimi jeziki za označevanje, ki bazirajo na formatu XML. Slogi CSS omogočajo, da zgled določenih tipov elementov definiramo le enkrat, zato po definiranju zgleda tega ni treba narediti za vsak element posebej. Najnovejša različica je CSS 3, vsebuje pa podporo za animacije in prehode ter podporo za prilagajanje velikosti oken (css file extension, 2018).

Sintaktične jezikovne predloge oziroma SASS (ang. Syntactically Awesome Style Sheets) so nadgradnja slogov CSS, saj nadgrajujejo njihova pravila in opise in tako naredijo kodo še močnejšo. Dodane so spremenljivke, pogojni izrazi, zanke in funkcije. Skriptni jezik SASS omogoča lažjo organizacijo datotek CSS ter hitrejši razvoj aplikacij. Datoteke z definiranimi pravili ne moremo uporabiti neposredno v dokumentu, saj jo je pred tem treba prevesti v jezik CSS, zato jezik SASS ne more delovati izven okvira jezika CSS. Namenjen je predvsem boljši strukturi, berljivosti in enostavnosti definiranja oblik elementov (SASS_REFERENCE, 2015).

1.4.2 Enostranske aplikacije

Enostranske aplikacije, v svetu računalništva poznane pod kratico SPA (ang. Single Page Application), so aplikacije, ki se generirajo na odjemalcu in se v brskalnik naložijo le enkrat (Mikowski & Powell, 2013).

Začetki razvoja enostranskih aplikacij segajo v obdobje dinamičnih spletnih strani, ko so se pojavile v obliki vstavljenih elementov in so za izvajanje potrebovale namestitev vtičnikov. Takrat se je tudi prvič pojavil JavaScript. Težave tovrstnih aplikacij so bile ravno z nameščanjem dodatkov, saj ni bilo mogoče zagotoviti, da bodo vsi uporabniki že imeli nameščene (Sušnik, 2017).

JavaScript je programski jezik, ki se je najprej pojavil z namenom razvoja spletnih strani, kjer se uporablja skupaj z jezikom HTML. Z JavaScriptom je omogočeno dinamično dodajanje, spreminjanje in odstranjevanje elementov dokumenta HTML. JavaScripta

deluje po principu asinhronega pristopa. Naloge se izvajajo zaporedoma, kar pomeni, da se šele po koncu prve lahko začne naslednja naloga. S svojim asinhronim modelom učinkovito rešuje težave, ki jih povzročajo dolgotrajne operacije. Največja revolucija se je zgodila leta 2008, ko je podjetje Google izdalo svoj brskalnik Chrome, ki je za izvajanje JavaScripta uporabljal nov V8-pogon. Ta je bil po hitrosti izvajanja za takratne čase nepredstavljivo hiter. Razvijalci so v sklopu tega videli nove načine, kako uporabiti to hitrost, in tako so nastale spletne aplikacije, ki jih uporabljamo danes (Haviv, 2014).

TypeScript je razširitev JavaScripta in je namenjen klasičnemu programiranju, kot ga uvajajo drugi objektni programski jeziki, na primer Java ali C#. Standardno sintakso dopolnjuje s statičnimi podatkovnimi tipi, kar pomeni, da po deklaraciji tipa spremenljivke ne moremo inicializirati z drugim podatkovnim tipom. Prinaša tudi podporo za generične funkcije, dedovanje, deklaracijo objektov in dekoratorje, ki omogočajo izvajanje operacij nad objekti izven neposrednega konteksta. TypeScripta ne smemo smatrati kot zamenjavo za JavaScript, saj predstavlja dodatek, ki pripomore k boljši strukturiranosti in berljivosti pri programiranju v JavaScriptu. Pred izvajanjem je tako kot jezik SASS treba tudi TypeScript prevesti v standardni JavaScript.

Napredek in poenotenje standardov brskalnikov sta počasi prišla na nivo, kjer se je zdelo smiselno vpliv JavaScripta razširiti na upravljanje in prikazovanje strani popolnoma neodvisno od generiranja s pomočjo strežnika. Pričelo se je obdobje enostranskih aplikacij, ki z višjo hitrostjo nalaganja zares izboljšujejo uporabniško izkušnjo. Ker so SPA samostojne aplikacije, lahko hitro postanejo zelo kompleksne, če se razvoja ne lotimo pravilno. To pa ima lahko vpliv tudi na hitrost izvajanja in velikost posamezne verzije aplikacije (Sušnik, 2017).

Tipična enostranska aplikacija je sestavljena iz posameznih komponent, ki se med delovanjem posamično osvežujejo ali spreminjajo. Pri nalaganju aplikacije se prenese tudi koda, ki skrbi za izvajanje aplikacije. Ob vsakokratni akciji nato aplikacija od strežnika zahteva nove podatke, ki se potem osvežijo na ustreznem delu strani. Takšna aplikacija osvežuje le tiste dele spletne aplikacije, kjer je prišlo do zahteve za spremembo, in tako ni potrebe po ponovnem osveževanju celotne strani (Valdarrama, 2014).

Prednost takšne aplikacije pred večstranskimi je predvsem v drugačni uporabniški izkušnji, saj ima uporabnik občutek, da uporablja namizno aplikacijo, saj se ne naloži ob vsaki spremembi (Valdarrama, 2014).

Prednosti in slabosti enostranskih aplikacij so tesno povezane s prednostmi in slabostmi, ki jih nosi generiranje spletnih strani na odjemalcu. Prednosti so naslednje (Valdarrama, 2014):

- hitro delovanje celotne aplikacije;
- ohranjanje zdravega stanja aplikacije skozi njen celoten življenjski cikel;

- logika odjemalca je povsem ločena od logike zaledja.

Slabosti pa so sledeče (Valdarrama, 2014):

- z velikostjo aplikacije se čas začetnega nalaganja podaljšuje;
- aplikacija na strani odjemalca lahko zahteva izvajanje poslovne logike, ki je ne bi smeli razkriti;
- nezdružljivost z brskalniki, ki ne podpirajo JavaSripta ali pa je ta onemogočen.

1.4.3 Angular

Angular predstavlja ogrodje za izdelavo enostranskih aplikacij na osnovi spletnih tehnologij. Prva verzija je bila predstavljena leta 2009. Vse dokler Google ni prevzel projekta, je bila njegova popularnost nizka, od takrat naprej pa prejema zelo veliko pozornosti. Njegova popularnost je trenutno daleč pred drugimi ogrodji (Hannah, 2015).

Predstavlja skupek najboljših praks za izdelavo spletnih aplikacij, ki jih je možno enostavno vzdrževati. Ogrodje se močno opira na knjižnico RxJS, saj delo s tokovi poenostavlja sledenje načelom SPA. Osredotoča se predvsem na podporo izdelavi spletnih in tudi mobilnih aplikacij. Uporabimo ga lahko tudi za razvoj katere koli druge vrste aplikacij. Bistvo Angularja je modularnost. Ogrodje je povsem ločeno od končne platforme, za komunikacijo z njo pa izrablja izrisovalnik (Sušnik, 2017).

Ogrodje uporablja več različnih konceptov, ki so poznani v svetu programiranja. Koncepti, ki jih uporablja, so naslednji (Green & Shyam, 2013):

- Generiranje zgleda na strani odjemalca s pomočjo predlog. Predloge so vnaprej pripravljeni deli spletne aplikacije, ki omogočajo, da razvijalec lažje generira dele spletne strani, ki so vizualno identični, vendar nosijo drugačne podatke. Angular popelje funkcionalnost predlog še korak dlje. Z njimi sestavljamo tudi osnovne dele aplikacije, kot so na primer celotni pogledi. V njih napišemo tudi oznake, ki predstavljajo, kako se bo del spletne strani obnašal ob delovanju.
- Uporaba arhitekture MVC, pri kateri so vsi trije koncepti (model, kontroler in pogled) jasno ločeni. Model predmeta dokumenta (DOM – ang. Document Object Model), ki predstavlja trenutno stran, v Angularju predstavlja »pogled«.
- Vezava podatkov, ki jih pri klasičnem razvoju spletnih aplikacij vrinemo v DOM. Če se podatki spremenijo, moramo počistiti mesto starih podatkov in nato ponoviti postopek vrivanja novih podatkov. Ta postopek je precej klasičen in nam lahko včasih predstavlja kar nekaj nepotrebnega dela. Angular ta problem rešuje avtomatično; komponenti povemo, katere spremenljivke objektov predstavljajo podatke, ki jih morajo prikazovati. Komponenta nato sama ve, kako prikazovati podatke in kdaj se osvežiti, če pride do spremembe podatkov.

- Injektiranje komponent nam omogoča, da v komponenti le povemo, katere druge komponente potrebujemo za delovanje. Angular nato uredi, da naša komponenta dobi instanco želene komponente.
- Omogoča nam pisanje novih direktiv, kar pomeni, da lahko našo kodo HTML dopolnimo z novimi direktivami, ki jih drugače ne najdemo v jeziku HTML. Angular bo prepoznal, za katero direktivo gre, in ji dodal ustrezno logiko. Tako pohitri razvoj nekaterih delov aplikacije.

Angular nudi širok nabor rešitev, ki lahko razvoj spletnih aplikacij bistveno olajšajo. Vendar pa prihaja tudi do naslednjih težav (Hannah, 2015):

- Največji problem je hitrost delovanja. Pri manjših aplikacijah je delovanje hitro, pri večjih aplikacijah pa se hitrost drastično zmanjša. Krivec za nižjo hitrost je samodejno osveževanje komponent. Reševanje teh težav zahteva dobro znanje programskega jezika, kar pa lahko predstavlja strošek.
- Problematična je tudi nova verzija ogrodja; verzija 2 bo namreč vsebovala spremenjeno strukturo delovanja aplikacije, kar pomeni, da ne bo združljiva s prejšnjimi verzijami. Hkrati to pomeni, da se bo treba naučiti novega sistema dela.
- Angular ima veliko funkcij in pa unikaten način dela, zato njegovo spoznavanje za mnoge prestavlja veliko dodatnega učenja.

1.5 Kaj lahko pričakujemo v prihodnosti?

Pomena in prednosti poslovanja preko interneta se podjetja čedalje bolj zavedajo, žal pa večini primanjkuje znanja. Če želimo doseči konkurenčno raven v EU, na tem področju torej ne smemo pozabiti na vlaganje v človeški kapital.

Rast e-poslovanja se pričakuje tudi v prihodnosti. Vse več podjetij uporablja elektronske pripomočke za lajšanje vsakodnevnih nalog (Krum, 2016). Direktorji, ki so še nedavno tega želeli imeti vse podatke o uspešnosti podjetja na računalniških zaslonih, sedaj vse te podatke zahtevajo na mobilnih zaslonih z možnostjo takojšnega obveščanja in samostojnega generiranja podatkov, ki jih potrebujejo v danem trenutku (Podjetje za računalniški inženiring, 2018).

Za primer vzemimo velike nakupovalne centre – kaj hitro se lahko zgodi, da ti ne bodo več potrebni. Nakupovanje predstavlja čas, vsi pa vemo, da je čas tudi denar, zato se vse več potrošnikov spogleduje z alternativami nakupovanju v trgovinah – to so spletni nakupi. Kupec izdelek naroči, če pa mu ne ustreza, ga lahko brez dodatnih stroškov pošlje nazaj (Kochhar, brez datuma).

E-poslovanje bo v prihodnosti ponujalo različne možnosti za personalizacijo, podjetja pa bodo lahko izbirala med množico ponudnikov; to bo zopet predstavljalo izziv, saj se bo

treba odločiti za pravo programsko rešitev. Odnos ponudnika rešitve s stranko pa bo tukaj ključnega pomena (Kochhar, brez datuma).

Prihodnost se že kaže tudi v mobilnem poslovanju. Mobilno poslovanje predstavlja brezžično e-poslovanje, kjer se uporablja mobilni telefon ali tablični računalnik. Glavni udeleženci pri m-poslovanju so B2C in B2B. Ker je tehnologija mobilnih aparatov in prenosa podatkov trenutno na visoki stopnji razvoja, se pričakuje tudi vse več mobilno podprtih aplikacij (Brewer, 2018).

Komunikacijska tehnologija kratkega dosega (NFC – ang. Near Field Communication) se v veliki meri že uporablja. Še vedno pa je možno tehnologijo še bolj približati končnim uporabnikom. Tehnologija NFC ni uporabna samo za plačevanje v trgovinah, ampak lahko ponudi veliko več. Z aplikacijo za pametne telefone lahko na primer pridobimo nove programe za pranje perila, nato pa te nove posodobitve preko pametnega telefona naložimo na pametni pralni stroj (Wong & Anand, 2013).

V zvezi s prihodnostjo kljub temu vedno obstaja element negotovosti, saj lahko nenadne motnje spremenijo njen celoten potek. Inovativna tehnologija bo še vedno spodbujala spremembe in ustvarjala nove načine, ki bi uporabnikom olajšali poslovanje. Podjetja se morajo odzvati na prihajajoče trende in se še naprej prilagajati hitrim spremembam na trgu (Kochhar).

2 METODOLOGIJE RAZVOJA INFORMACIJSKEGA SISTEMA

Informatika, ki je ustrezno razvita in vpeta v podjetje, na trgu predstavlja konkurenčno prednost. Metodologije razvoja informacijskih sistemov prav zaradi tega stremijo h gradnji takšnih sistemov, ki podjetju, ki nastopa na trgu, prinašajo konkurenčno prednost pred ostalimi podjetji (Kovačič, 1998, str. 40).

Informacijski sistem je sestavljen iz množice ljudi, strojev, postopkov in aktivnosti, ki omogočajo pridobivanje in uporabo informacij. Na podlagi teh informacij se nato uporabniki informacijskega sistema odločajo glede aktivnosti v podjetju. Informacijski sistem omgoča shranjevanje podatkov, njihovo obdelavo ter preoblikovanje v informacije, te pa potem zaposlenim omogočajo sprejemanje odločitev ter izvajanje poslovnih procesov znotraj podjetja (Gradišar, Jaklič, Damij & Baloh, 2005, str. 40–41).

Rupnik (brez datuma, str. 43) informacijski sistem definira kot množico komponent, ki so med seboj povezane in odvisne. To so strojna in programska oprema ter komunikacijska oprema in ljudje. Zgoraj naštete komponente zbirajo, procesirajo, hranijo ter porazdeljujejo podatke in s tem zagotavljajo podporo odločitvam v procesih organizacije.

Vsak informacijski sistem je zasnovan tako, da čim bolj poenostavi in odpravi tiste naloge, ki so vnaprej znane oziroma pričakovane. Prenova informacijskega sistema z že

poenostavljenimi nalogami je zahtevno ter dolgotrajno delo, saj mora končnemu izdelku zagotoviti visoko stopnjo integritete in robustnosti kot tudi sprejetost med njegovimi uporabniki (Leau, Loo, Tham & Tan, 2012, str. 1).

Za uspešno prenovo poslovnih procesov podjetja je ključna uporaba metodologije razvoja informacijskih sistemov, katerih uporaba omogoča skrajšanje razvojnega časa in lažje vodenje ter obvladovanje projekta, obenem pa prispeva tudi k zadovoljstvu uporabnikov po končni implementaciji rešitve (Zornada, 2002, str. 223–236).

Razvojna metodologija informacijskega sistema zajema postopke in tehnike, ki se uporabljajo za strukturiranje, načrtovanje in nadzor nad razvojem informacijskega sistema (Software Development Methodologies, 2018).

Metodologijo načrtovanja in izgradnje informacijskih sistemov opredeljujejo tudi Gradišar, Jaklič, Damij in Baloh (2005, str. 267–268), in sicer je poimenovana kot skupek tehnik, orodij, postopkov ter drugih dokumentacijskih pripomočkov, ki jih razvijalci uporabljajo pri načrtovanju in implementaciji. Ti omogočajo tudi upravljanje, kontroliranje in vrednotenje projektov izgradnje in informatizacije informacijskega sistema. Metodologija je sestavljena iz faz in njihovih podfaz, razvijalec sistema pa lahko izbere najprimernejšo tehniko za vsako fazo posebej.

2.1 Delitev metodologij

Teorija je polna različnih delitev metodologij. Gradišar, Jaklič, Damij in Baloh (2005, str. 268–269) metodologijo razvoja informacijske rešitve delijo v naslednje kategorije:

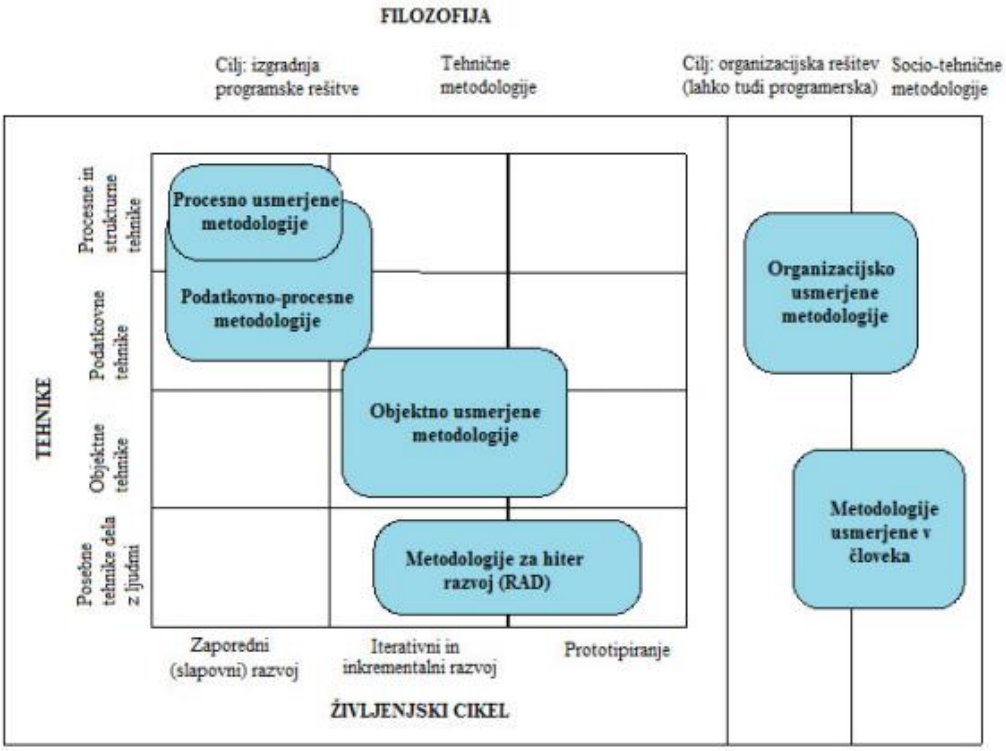
- Tradicionalna metodologija pri razvoju informacijskih sistemov uporablja vnaprej definirane razvojne faze, zato jih lahko imenujemo tudi zaporedne, saj si zaporedoma sledijo. Proces razvoja je tako linearen, saj se vsaka naslednja faza lahko začne šele po zaključku predhodne faze.
- Prototipni pristop uporablja metode in tehnike tradicionalnih metodologij, kar pomeni, da se najprej razvije poenostavljena rešitev, ki je brez specifičnih posebnosti, in se jo nato dopolnjuje oziroma po potrebi tudi zavrže. Razvijalci na ta način predstavijo informacijski sistem, ki ga nato s uporabniki sproti dopolnjujejo oziroma spreminjajo.
- Objektni pristop obravnava podatkovni in procesni vidik razvoja modela kot skupek. Primer objektnega pristopa je tabelarni razvoj aplikacij (TAD – ang. Tabular Appliactaion Development).

Krisper in drugi (2003) v svojem delu metodologijo razvoja informacijski sistemov delijo v tri skupine glede na tip metodologije, utežitev metodologije in težo metodologije.

2.1.1 Delitev metodologij glede na tip metodologije

Metodologije razvoja programske opreme se glede na njihov tip delijo na podlagi filozofije. Upoštevani so cilji metodologije in sociološka komponenta. Tako so metodologije razdeljene v dve kategoriji, in sicer poznamo socio-tehnične, kjer je cilj organizacijska rešitev, ter tehnične, ki se na podlagi življenjskih ciklov še dodatno delijo, kot je prikazano na Sliki 2 (Krisper, in drugi, 2003).

Slika 2: Delitev glede na tip metodologije



Vir: Krisper in drugi (2003).

Tako se metodologije razvoja informacijskih sistemov glede na tip delijo v dve kategoriji (Krisper in drugi, 2003):

- Socio-tehnične metodologije so organizacijsko in uporabniško usmerjene metodologije.
- Tehnične metodologije predstavljajo procesno usmerjene, podatkovno-procesne in objektno usmerjene metodologije ter metodologije za hiter razvoj.

Socio-tehnične metodologije se ne osredotočajo le na tehnično plat razvoja sistema, ampak pri razvoju upoštevajo tudi ljudi, ki bodo s sistemom delali, zato je lahko cilj takšnih metodologij tudi organizacijska rešitev. Metodologije, ki so bolj usmerjene v organizacijsko rešitev, uvrščamo med organizacijsko usmerjene metodologije, metodologije, usmerjene v sociološko komponento, pa med metodologije, usmerjene v človeka (Krisper in drugi, 2003):

- Organizacijsko usmerjene metodologije se osredotočajo na modeliranje organizacije kot celote. Tako je lahko v teoriji njihov cilj tudi organizacijska rešitev, čeprav je ta skoraj vedno razvita poleg novo razvite programske rešitve, ki nato omogoča uporabo na novo modeliranega organizacijskega modela. Zaradi upoštevanja sociološke komponente se metodologije te vrste pogosto prepletajo z metodologijami, usmerjenimi v človeka.
- Metodologije, usmerjene v človeka, za razvoj programskih rešitev uporabljajo socio-tehnični pristop, saj je upoštevana tudi tehnološka komponenta. Te metodologije so nastale kot kritika tehnične metodologije, saj ta zanemarja sociološki vidik. Značilnost takih metodologij je, da pri načrtu sodelujejo vsi uporabniki končne programske rešitve. Tako so vključeni vsi neposredni in posredni uporabniki rešitve. Zaradi širokega kroga ljudi, ki so vpeti v odločanje, ta metodologija vključuje tudi lastnosti organizacijsko usmerjenih metodologij.

Pod tehnične metodologije spadajo tehnologije, ki imajo za cilj izgradnjo programske rešitve. Te na podlagi uporabljenih tehnik in življenjskega cikla delimo v naslednje štiri skupine (Krisper in drugi, 2003):

- Procesno usmerjene metodologije – te so osredotočene na modeliranje procesov v organizaciji, pri tem pa so v uporabi procesne tehnike. Tehnike so lahko odločitvena drevesa, delovni tokovi ali akcijski diagrami. Uporabljeni so lahko tudi strukturni diagrami, s katerimi so prikazane strukture sistema. Vsem zgoraj naštetim metodologijam je največkrat skupno to, da metodologije uporabljajo zaporedni (slapovni) življenjski cikel z modifikacijami.
- Podatkovno-procesne metodologije – gre za nadgradnjo procesno usmerjenih metodologij, saj vključujejo tudi modeliranje podatkov v sistemu. Uporabljene tehnike vključujejo tako procesne in strukturne kot tudi tehnike podatkovnega modeliranja. Enako kot procesno usmerjene metodologije tudi podatkovno-procesne metodologije uporabljajo zaporedni (slapovni) življenjski cikel z modifikacijami.
- Objektno usmerjene metodologije – slednje se od zgoraj omenjenih razlikujejo, saj se osredotočajo na modeliranje objektov, ki so osnovni gradniki informacijskega sitama, in hkrati obravnavajo podatke in procese. Modeliranje objektov zahteva drugačen pristop analize, načrtovanja ter uporabe objektno usmerjenih programskih jezikov. Metodologije uvajajo poenotenje razvoja programske opreme, saj se s skupnimi koncepti srečujejo skozi celoten proces razvoja. Skupni procesi omogočajo možnost ponovne uporabe programske kode. V objektno usmerjenih metodologijah sta za razvoj informacijskega sistema uporabljena iterativni in inkrementalni življenjski cikel sistema. Tak sistem se zgradi s preizkušenimi gradniki in večjim številom iteracij.
- Metodologije za hiter razvoj – ta vrsta je nastala zaradi tehnoloških sprememb, ki spreminjajo zahteve informacijskih sistemov. Tudi te metodologije temeljijo na iterativnem življenjskem ciklu in omogočajo hiter razvoj programske opreme. Zgrajene

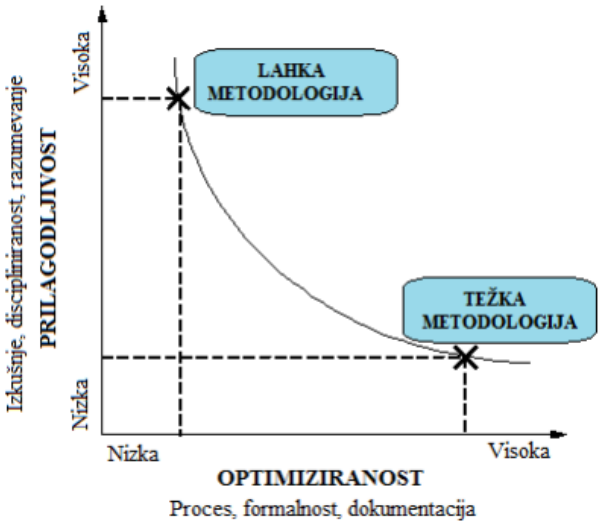
so z orodji za hiter razvoj programske opreme, poudarek pa je na implementaciji in testiranju.

2.1.2 Delitev metodologij glede na težo metodologije

Zaradi kritike obsežnih formalnih metodologij in novih pristopov k razvoju programske opreme se je pojavila nova delitev; metodologija se je začela deliti na težko in lahko. Teža se določa glede na obseg in gostoto, kar prikazuje tudi Slika 3. Gostota definira podrobnosti oziroma formalnost. Metodologije z višjo gostoto so bolj formalne in njihovi elementi podrobneje definirani kot pri metodologijah z nižjo gostoto. Pri slednjih so interpretacije prepuščene posamezniku. Drugi element, ki definira metodologijo, je njen obseg. Ta je določen na podlagi števila različnih elementov; ti elementi so aktivnosti, vloge, izdelki in pa standardi (Krisper in drugi, 2003).

Highsmith (2002, str. 202–203) v svoji knjigi ugotavlja, da se je treba pri delitvi glede na težo osredotočiti na tri dejavnike, in sicer težo postopka v primerjavi z izkušnjami, težo formalnosti v primerjavi z disciplino in težo dokumentacije v primerjavi z razumevanjem. Pri določanju teže se tako primerja skupek vseh dejavnikov ter vpliv na metodologijo, vse skupaj pa se odraža kot primerjava med optimiziranostjo in prilagodljivostjo metodologije.

Slika 3: Določitev teže metodologije



Vir: Highsmith (2002).

Med težke metodologije uvrščamo metodologije z večjim obsegom in gostoto. V tem primeru sta tudi stopnji formalizacije in optimizacije na visokem nivoju. Značilna je natančno pripravljena dokumentacija ter lastnost, da vsebuje podatke o predpisanih orodjih in tehnikah. Takšna metodologija je visoko optimizirana, zato se uporablja za izgradnjo kritičnih programskih rešitev, pri katerih sodelujejo večje razvojne skupine (Krisper in drugi, 2003).

Za večje projekte sta potrebni dodatna formalnost in obsežna dokumentacija, kar pomeni, da je za pripravo le-teh potrebnega veliko znanja in izkušenj, še ugotavlja Higsmith (2002, str. 203).

Težke metodologije so v uporabi, kadar (Krisper in drugi, 2003):

- je cilj razvoj visokokritičnega sistema z zahtevano ustrezno dokumentacijo,
- so zahteve dobro definirane in stabilne,
- je cilj prenova organizacijskega in razvoj novega programskega sistema,
- so zaradi manj izkušenih razvijalcev točno opredeljena moralna pravila, ki nadomeščajo njihovo znanje in izkušnje,
- je formalnost na visokem nivoju.

Med lahke metodologije spadajo tiste, ki imajo manjši obseg in manjšo gostoto. Pri lahkih metodologijah je poudarek na sodelovanju med udeleženci projekta in na razvoju programske rešitve, manj pa sta upoštevana sociološki in poslovni vidik. Zaradi tega je stopnja prilagodljivosti teh metodologij visoka, to pa pomeni, da se hitro odzivajo na okoljske zahteve. Temelj so izkušnje, disciplina in razumevanje, zato so primerne za manjše skupine z nižjo stopnjo formalnosti. Lažja kot je metodologija, manj je potrebnega optimiziranja in prilagajanja (Krisper in drugi, 2003).

Lahke metodologije se uporabljajo, ko (Krisper in drugi, 2003):

- je cilj razvoj programske rešitve,
- so razvijalci izkušeni, odgovorni in motivirani,
- je stranka pripravljena sodelovati,
- so zahteve za programsko rešitev spreminjajoče se in nepredvidljive,
- se razvija majhen sistem z nizko stopnjo kritičnosti, ki ga je mogoče razviti z malo razvojniki.

2.1.3 Delitev metodologij glede na utežitev metodologije

Zaradi dejstva, da je razvoj programske opreme sestavljen iz zaporednega izvajanja različnih aktivnosti, se je razvila delitev metodologij glede na utežitev. Razvoj programske opreme se začne s pripravo analize in načrta sprememb, nato sledi razvoj in na koncu še test programske opreme. Glede na poudarjenost aktivnosti v metodologiji lahko te razdelimo na spredaj utežene, pri katerih sta poudarjeni prvi dve aktivnosti, ter na zadaj utežene metodologije s poudarkom na zadnjih dveh aktivnostih. O uravnoteženi metodologiji govorimo, kadar je pri razvoju nekaterih delov informacijskega sistema poudarek na prvih dveh aktivnostih, pri drugih pa sta poudarjeni zadnji dve aktivnosti. Spredaj utežene metodologije so znane kot težke metodologije, zadaj utežene pa kot lahke metodologije (Krisper in drugi, 2003).

Pri spredaj uteženih je poudarek na analizi in načrtu, kar pomeni, da so sistemske zahteve podrobno analizirane ter načrt izgradnje natančno določen. Dodatne analize in načrti se lahko izvajajo tudi kasneje, vendar se načrt ne spreminja več kaj dosti. Zaradi dobro izdelanih analiz in načrtov je kodiranje rutinsko, manj pa je tudi končnega testiranja, saj dobro definirani načrti zmanjšajo možnosti za napake (Krisper in drugi, 2003).

S spredaj uteženimi metodologijami se razvija (Krisper in drugi, 2003):

- kritične sisteme, kjer so predvideni različni alternativni scenariji dogajanja, ki so tudi zajeti v načrtu;
- obsežne in kompleksne sisteme, ki so namenjeni velikim razvojnim skupinam in pri katerih so potrebni podrobnejši načrti za lažjo razdelitev dela med člani;
- sisteme s stabilnimi zahtevami, kjer so zahteve dobro specificirane in stabilne ter kasneje ne prihaja do večjih sprememb.

Večina klasičnih metodologij poudarja pomembnost analize in načrta, zato jih uvrščamo med spredaj utežene. Ker je za pripravo podrobne analize in načrta potrebnega veliko časa, končni rezultat pa je lahko zaradi tega tudi neuporaben produkt, ker se zahteve vse hitreje spreminjajo, je to tudi slabost spredaj uteženih metodologij (Krisper in drugi, 2003).

Zadaj utežene metodologije poudarjajo postopek izvedbe in testiranja, zato se pri postopku analize in načrtovanja ne ukvarjajo toliko s podrobnostmi. Navadno slednja dva potekata skupaj s kodiranjem, po končani izvedbi pa se izvede temeljit test, s katerim se ugotavlja morebitne potrebne popravke kode. Na koncu nastane stabilen izdelek s pomanjkljivostjo manj robustne arhitekture izdelka. Največja kritika takšne metodologije je, da ni primerna za razvoj obsežnih sistemov, sem pa uvrščamo metodologijo prototipiranja (Krisper in drugi, 2003).

Z zadaj uteženimi metodologijami se razvija (Krisper in drugi, 2003):

- manjše sisteme z izkušenimi razvojnimi skupinami – izkušenost članov nadomešča slabšo analizo in načrt;
- sisteme s slabo določenimi zahtevami, ki se bodo še spreminjale;
- nekritične sisteme, saj slabša analiza in načrt nista primerna za kritične sisteme;
- sisteme, v katerih je uporabljena nepoznana tehnologija – tako lahko preizkusimo in spoznamo delovanje.

Uravnotežene metodologije poudarjajo, da za dele sistema, katerih zahteve so nam znane in so stabilne, izvedemo podrobno analizo in načrt, za druge dele sistema, ki so nam nepoznani oziroma zahteve niso stabilne, pa takoj začnemo s kodiranjem in testiranjem. Tako kombiniramo oba pristopa, kar ni nujno vedno dobra odločitev, saj obstajajo procesi, ki nimajo vseh značilnosti spredaj ali zadaj uteženih procesov (Krisper in drugi, 2003).

Delitev metodologij pri razvoju informacijskih sistemov avtorji navajajo različno, vendar večino izmed njih lahko razdelimo na tradicionalne (težke) in agilne (lahke). V zadnjem času so v ospredju agilne metode, ki pa niso primerne za vse oblike razvoja sistemov, zato tradicionalnih metodologij ne bodo mogle povsem nadomestiti.

2.2 Tradicionalne metodologije

Razvoj informacijskega sistema je določen s sosledjem faz, o čemer govori tudi Bajec, saj nam pove, kako si sledijo faze v razvoju informacijskega sistema (Bajec, brez datuma, str. 17). Proces izgradnje sledi sosledju določenih faz v življenjskem ciklu razvoja sistema. Poleg analize, izgradnje in testiranja vključuje različne faze življenjskega cikla (Yu, Wooi, Wai & Soo, 2012).

Vsak nov razvoj informacijskega sistema oziroma menjava generacije, za kar šteje nova različica programske opreme, sledi fazam življenjskega cikla razvoja sistema. Zaključek vsake faze je definiran s časovnim mejnikom, nato pa sledi sprejem novih odločitev. Življenjski cikel razvoja informacijskega sistema je sestavljen iz naslednjih faz (Krisper in drugi, 2003):

- Začetna faza določa obseg projekta in je namenjena iskanju zunanjih entitet, s katerimi bo sistem sodeloval, ter določanju kriterijev, po katerih se bo ocenjevalo uspešnost projekta. Pripravi se ocena tveganja ter fazni načrt z vsemi časovnimi mejniki projekta.
- Faza zbiranja informacij se uporablja za analiziranje problemskega področja, izdelavo projektnega plana in razširitev tveganih elementov. V tej fazi se izdela tudi načrt izgradnje.
- Faza konstrukcije obsega izdelavo informacijskega sistema, ki je bil definiran v načrtu izgradnje. V tej fazi se izvede tudi testiranje in pripravi uporabniška navodila.
- Faza prevzema pomeni implementacijo novega sistema ali programske opreme v obstoječ sistem oziroma predajo naročniku. V tej fazi se na strani naročnika lahko pojavijo potrebe po popravkih ali dopolnitvah.

Podobno definicijo predstavlja tudi Bajec (brez datuma, str. 17), ki ugotavlja, da razvoj informacijskega sistema, ki sledi določenemu življenjskemu ciklu, ki določa zaporedje faz razvoja, ni nič drugačen od večine razvojnih procesov. Tudi tukaj so faze razvoja definirane kot analiza, načrtovanje, testiranje, implementacija, uvedba in vzdrževanje.

Tradicionalne metodologije že na začetku življenjskega cikla zahtevajo pripravo podrobne dokumentacije, ki se ji sledi skozi celoten življenjski cikel. Te se uvrščajo med težke metodologije, saj sledijo zaporedju korakov, ki se začne z analizo rešitve, sledi njena izgradnja, nato pa še testiranje in implementacija (Yu, Wooi, Wai & Soo, 2012).

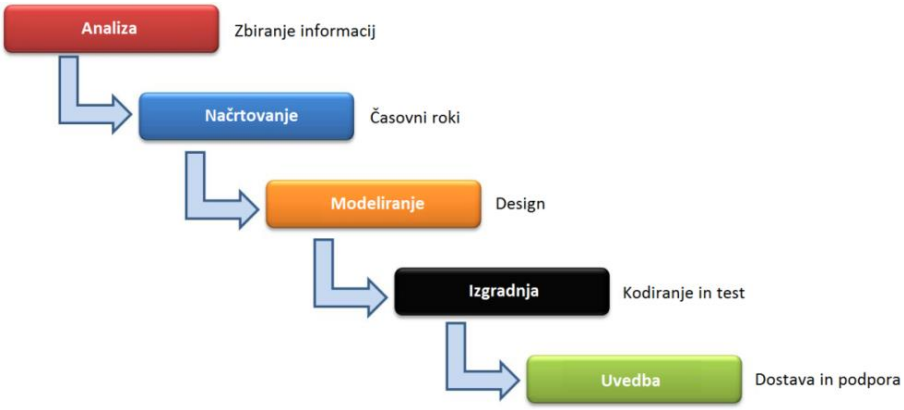
Iz tradicionalne metodologije, ki temelji na življenjskem ciklu, so se razvili modeli, ki so prikazani v nadaljevanju. Praksa je pokazala, da se najpogosteje uporablja kombinacija

različnih modelov oziroma izpeljank iz zaporednega modela, ki temelji na sledenju posameznih aktivnosti.

2.2.1 Zaporedni oziroma slapovni model

Zaporedni oziroma slapovni model je model, pri katerem razvojne faze informacijskega sistema potekajo po vnaprej določenem zaporedju. Model je prikazan na Sliki 4. Naslednja faza se začne šele po končani predhodni fazi, kar pomeni, da se faze med seboj ne prekrivajo (Hvala 2011, str. 30).

Slika 4: Zaporedni oziroma slapovni model



Vir: Chowdhury, Bhowmik, Hasan & Rahim (2018).

Zaporedni oziroma slapovni model je najhitrejši razvojni model, pri katerem si faze analize, načrtovanja, izvedbe in uvedbe sledijo zaporedno in vračanje nazaj po lestvici ni več mogoče. Ta model je značilen za prve oblike strukturnega pristopa. Uporablja se za obsežne in kompleksne projekte z že dobro definiranimi zahtevami, ki se med projektom ne bodo več bistveno spreminjale. Zaradi obsežne dokumentacije model omogoča natančno vodenje in pregled nad projektom (Bajec, brez datuma, str. 22).

Metodologija življenjskega cikla razvoja informacijskega sistema s slapovnim modelom je sestavljena iz naslednjih faz (Gradišar, Jaklič, Damij & Baloh, 2005, str. 270–272):

- Študija izvedljivosti je prvi korak razvoja informacijskega sistema, s katerim se preveri s strani naročnika sporočene zahteve in morebitne probleme in pripravi možne rešitve, koristi ter določi stroške in koristi nove rešitve.
- Raziskava sistema je naslednja faza, v kateri se preveri omejitve in probleme obstoječega sistema ter na podlagi tega pripravi specifikacije in analize.
- Načrtovanje spremeni logični model, ki je bil pripravljen v predhodni fazi, v načrt informacijskega sistema, pri tem pa se definira tudi način zajema podatkov ter vhodne in izhodne podatke sistema.

- Implementacija je faza, v kateri se zgradi sistem ter naredi prehod s starega na novi sistem in izobrazi osebje za uporabo novega sistema.
- Vzdrževanje sistema temelji na življenjskem ciklu razvoja sistema in se izvaja po tem, ko je bil prehod na novi sistem že izveden. V tej fazi se sprotno odpravljajo odkrite napake in urejajo prilagoditve sistema, kar zagotavlja učinkovito delovanje novega sistema.

Ključna prednost slapovnega modela je predvsem manjša poraba režijskega dela, ki ni neposredno povezana z izdelavo nove programske opreme (vodenje projekta), ker je vse načrtovanje v celoti izvedeno vnaprej (Bajec, brez datuma, str. 23–24). Visoka stopnja formalizacije slapovnega modela skoraj vedno zagotavlja kakovost končnega izdelka na visoki ravni (Hvala, 2011, str. 30).

Slabost slapovnega modela je predvsem njegova fleksibilnost, saj vsaka sprememba zahtev povzroči veliko dodatnega napora za ponovne analize. Ker pa se informacijsko okolje zelo hitro spreminja, zahteve nikoli niso statične. Med slabosti lahko uvrstimo tudi dejstvo, da se nova faza lahko začne šele z zaključkom predhodne in da vračanje nazaj ni več mogoče, kar pa je v praksi skoraj neizvedljivo. Ker vračanje k predhodnim fazam vse do konca projekta ni mogoče, se lahko zgodi, da razviti sistem ne bo ustrezal dejanskim zahtevam naročnika (Rupnik, brez datuma, str. 77).

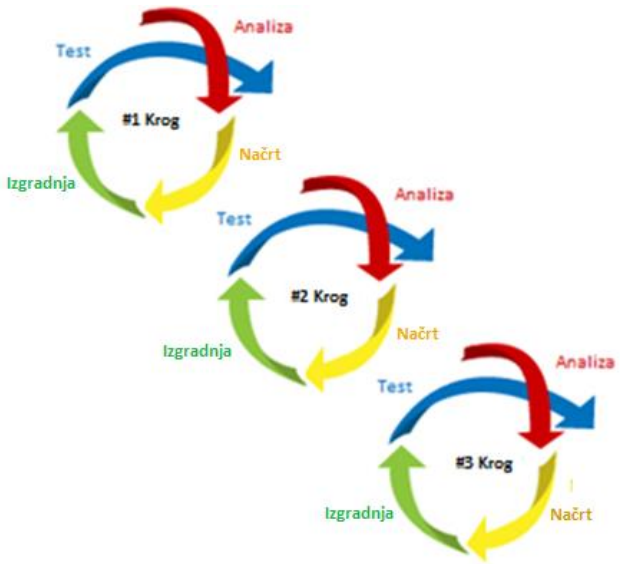
Podobno slabost izpostavlja tudi Hvala (2011, str. 30), ki navaja nezmožnost prilagajanja razmeram na trgu ter preveliko količino dokumentacije, ki spremlja vsako fazo projekta.

2.2.2 Iterativni model

Iterativni modeli so se razvili zaradi pomanjkljivosti slapovnega modela, saj ta ne omogoča vračanja na predhodne faze in njihovega ponavljanja. Značilnost iterativnih modelov je postopen razvoj, saj se posamezne faze ne končajo v celoti, ampak le delno, pri tem pa se celoten cikel ponavlja, dokler projekt ni končan. Najbolj prepoznaven iterativni model je spiralni model (Krisper in drugi, 2003). Pri spiralnem modelu so elementi načrtovanja, prototipiranja in iteracij združeni in jih ponavljamo toliko časa, dokler projekt ni končan (Awad, 2005, str. 6).

Iterativni model poteka v več iteracijah oziroma se izvaja več zaporednih aktivnosti hkrati. Izvedene so na podlagi načrta in se končajo z realizacijo izdelka. Kot je razvidno s Slike 5, gre vsaka posamezna iteracija navadno čez vse faze življenjskega cikla razvoja informacijskega sistema (analiza, načrt, izgradnja, uvedba). Pri vsaki se razvije določen del funkcionalnosti celotnega sistema, pri čemer najprej razvijemo najbolj tvegane dele sistema. Proces razvoja je prikazan na spodnji sliki. Med izvajanjem iteracij se ne sprejema sprememb zahtev, izvajanje vsake pa traja od sedem do štirinajst dni, da se izvede od začetka do konca. Ko se uspešno zaključi prva iteracija, se na osnovi njenih rezultatov določi naslednjo in začne z njeno izvedbo (Bajec, brez datuma, str. 25–28).

Slika 5: Iterativni model



Vir: 360logica Software Testing Services (2015).

Prednosti iterativnega modela so, da se najbolj tvegane dele projekta rešuje na začetku in se zato lažje kontrolira vrednost investicije, poleg tega pa iteracija omogoča predčasno predajo posameznega dela projekta. S tem se zagotovi povratne informacije s strani uporabnikov, ki jih nato upoštevamo pri nadaljnjih iteracijah. Ciljni mejniki so posamezne iteracije in tako lažje spremljamo izvedeno delo, saj se osredotočamo na eno iteracijo in ne na celoten projekt razvoja (Rupnik, brez datuma, str. 82).

Slabost iterativnega modela je predvsem težko in nepredvidljivo načrtovanje projekta. Nemogoče je vnaprej predvideti, koliko posameznih iteracij bo potrebnih za razvoj končnega produkta. Posledica tega pa je zahtevno vodenje projekta (Bajec, brez datuma, str. 31).

2.2.3 Prototipni model

Prototipni model je izpeljanka iz iterativnega modela. Temelji na izdelavi prototipov, ki so izdelani predhodno. Prototipi so še nepopolne različice sistema (Bajec, brez datuma, str. 32), ki jih lahko uporabljamo na dva načina. Prvi način je uporaba kot specifikacija sistema, s katero se pridobi podobo sistema, ki ga želimo razviti, in se v nadaljevanju ta prototip zavrže. Drugi način pa je uporaba prototipa kot osnove za izdelavo končne verzije sistema, kar je prikazano tudi na Sliki 6 (Rupnik, brez datuma, str. 84).

Slika 6: Prototipni model

Vir: Sava (2017).

S prototipi se olajša komunikacija z uporabnikom, zato se jih uporablja v različnih fazah razvoja. Z naročnikom se najprej uskladi vse zahteve sistema, nato pa se na podlagi teh izdela prototip, ki se ga preda v testiranje uporabniku. Uporabnik preko povratnih informacij sporoči, kaj bi bilo treba popraviti in česa ne. Na podlagi prejetih informacij se izdela nov prototip ter se postopek ponavlja, dokler naročnik ne odobri delovanja sistema (Krisper in drugi, 2003).

Z zgoraj opisanim se strinja tudi Sava (2017), ki še navaja, da se model uporablja za velike in zahtevne sisteme, za katere je na začetku procesa težko definirati vse funkcionalnosti. V takšnih situacijah omogoča dostop do prototipa znaten prispevek k razumevanju in opredelitvi specifikacij.

Prednosti tega modela so naslednje (Sava, 2017):

- Uporabniki so neposredno vključeni v razvoj.
- Spodbuja uporabnike, da spremenijo svoje zahteve, ko je produkt še v prototipni fazi.
- Uporabniki bolje razumejo, kako sistem deluje, saj ga že prej testirajo.
- Napake je mogoče zaznati veliko prej.
- Povratne informacije uporabnikov so veliko hitrejše, kar vodi do boljše rešitve.
- Krajši čas izvedbe procesa in nižji stroški.

Model ima tudi pomanjkljivosti, ki so sledeče (Sava, 2017):

- Model lahko poveča zapletenost sistema.
- Slaba analiza projekta.
- Prekomerna poraba časa za razvoj prototipa.

2.2.4 Inkrementalni model

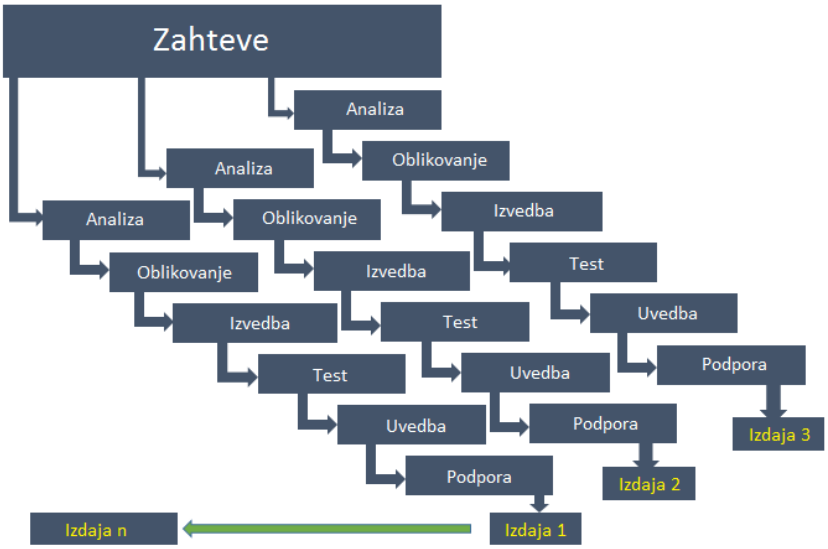
Pri inkrementalnem modelu gre za postopno gradnjo informacijske rešitve. Pri gradnji se omejimo na posamezen sklop in je ne razvijamo v celoti. Pri predaji uporabniku vse sklope povežemo v celoto (Bajec, brez datuma, str. 34–35).

Krisper in drugi (2003) ugotavljajo, da je inkrementalni model zelo pogosto uporabljen model razvoja informacijskega sistema, pri katerem se celotni modul razdeli na podmodule, v okviru katerih se nato rešujejo problemi neodvisno od ostalih. Podmoduli so deli funkcionalnosti celotnega sistema, ki so dovolj samostojni, da se lahko vključijo v produkcijo. Celoten projekt se tako razdeli na funkcionalna področja, kar je prikazano na spodnji sliki. Ta področja so posamezni inkrementi, ki se na koncu združijo v celoto.

Inkrementalni model ima to prednost, da naročnik lažje sledi napredku celotnega projekta, saj informacijsko rešitev prejema postopoma, sam pa tudi sodeluje pri testiranju in podaja povratne informacije za že razvite sklope (Bajec, brez datuma, str. 39). Prednost takšnega modela je tudi možnost zgodnjega odkrivanja napak in njihovo lažje odpravljanje, saj se vsak posamezni del končnega produkta preda v redno uporabo relativno zgodaj in ima uporabnik čas dodobra testirati prejeti del sklopa informacijskega sistema. Razvoj informacijskega sistema po inkrementalni metodi je v splošnem cenejši in manj tvegan od razvoja celotnega sistema v enem kosu (Krisper in drugi, 2003).

Slabost inkrementalnega modela je predvsem v tem, da ga ni moč uporabiti pri vseh projektih, saj nekaterih rešitev ni mogoče razdeliti in predati naročniku po delih. Lahko se tudi zgodi, da pri načrtovanju posamezni sklop informacijskega sistema narobe razporedimo, kar pa neugodno vpliva na izdelavo rešitve (Bajec, brez datuma, str. 39).

Slika 7: Inkrementalni model



Vir: *Naveen (2015).*

Kot je prikazano na Sliki 7, je inkrementalni model sestavljen iz kombinacije več slapovnih modelov, ki na koncu zagotovijo celovito aplikacijo. Vsak posamezno razvit modul predstavlja samostojno funkcijo in ga je končnim uporabnikom mogoče dostaviti za uporabo. Ker je vsak modul samostojna aplikacija in med moduli ni drugih odvisnosti, lahko verzijo dostavimo z začetno razvito funkcionalnostjo, druge module pa nato dostavljamo postopno z novimi verzijami (Naveen, 2015).

2.3 Agilne metodologije

V okolju, kjer se razvijajo informacijski sistemi in programske rešitve, so tehnološke spremembe stalnica. Takšno okolje je tudi zelo dinamično, konkurenca pa je na visokem nivoju. Zahtevno pa ni le okolje, ampak tudi kupci, ki veliko zahtevajo in pričakujejo produkt z dolgo življenjsko dobo. Da so podjetja na takšnem trgu lahko konkurenčna, se morajo uspešno odzivati na nepredvidljiva povpraševanja, kar pa z uporabo tradicionalnih razvojnih metodologij težko dosežejo. Zato so se začele razvijati agilne metodologije, ki rešujejo težave, ki jih povzroča neprilagodljivost tradicionalnih metodologij (Kettunen, 2009, str. 408).

Alternativni modeli razvoja, kot so prototipni razvoj, ekstremno programiranje, evolutivni razvoj, hiter razvoj aplikacij, objektni pristop, skupen razvoj aplikacij, inkrementalni razvoj in agilni razvoj, so se razvili zaradi informacijske tehnologije, ki je pri razvoju stremela k odpravljanju omejitev tradicionalnih metodologij razvoja informacijskega sistema. Alternativne metodologije nosijo to lastnost, da odpravljajo pomanjkljivosti tradicionalnih metodologij ter sledijo težnji po hitrejšem razvoju aplikacij, v katerega vključujejo tudi končnega kupca (Gradišar, Jaklič & Turk, 2007).

Agilni razvoj predstavlja način razmišljanja oziroma filozofijo, ki jo uporabljamo pri razvoju informacijskih rešitev. Agilne metode predstavljajo skupek posameznih elementov in priporočil, ki so sicer že zelo dolgo poznana, vendar so pri agilnih metodah združena v kombinacijo, ki ni poznana pri drugih metodologijah (Shore & Warden, 2008).

Agilne metodologije imajo to prednost, da se osredotočajo na hitrost in preprostost, tako da se lahko skupina, odgovorna za razvoj, osredotoča le na programiranje funkcionalnosti, potrebnih za izgradnjo delujočega proizvoda. Metodologije so agilne, ko (Abrahamsson, Salo, Ronkainen & Warsta, 2002):

- je uporabljen inkrementalni pristop pri razvoju, kar pomeni dostavo več manjših različic z več ponovitvami,
- razvojna ekipa in naročnik tesno sodelujeta,
- se metoda lahko hitro prilagaja kljub spremembam zahtev,
- je metoda na voljo v različnih razvojnih okoljih in je enostavna za uporabo.

Priznani znanstveniki s področja agilnih metodologij so se decembra 2001 sestali z namenom, da bi postavili temelje metodološkim osnovam agilnih oziroma tudi lahkih metodologij. Skupina 17 strokovnjakov je v Manifestu agilnega razvoja programske opreme (ang. The Agile Manifesto) ugotavljala skupne točke posameznih metodologij. Združeni v zvezo Agile Alliance so objavili nov trend lahkih metodologij, ki so znane po učinkovitosti in prilagodljivosti (Bajec & Krisper, 2003, str. 4).

Člani zveze so v manifestu definirali cilje v okviru iskanja boljših načinov za razvoj programske opreme; strinjali so se, da bodo pri razvoju sodelovali tudi sami ter pomagali ostalim. Poudarili so naslednje vrednote (Beck in drugi, 2001):

- Interakcije med posamezniki so pomembnejše od procesov in uporabljenih orodij.
- Delujoča programska oprema je pomembnejša od natančne dokumentacije.
- Stik s stranko je pomembnejši kot pa pogajanja v okviru pogodbe.
- Odzivnost na spremembe je bolj pomembna kot sledenje načrtu.

Prvo načelo interakcije med posamezniki poudarja, da stroga porazdelitev vlog ni smiselna, če za to nimamo ustreznih članov tima, da bi te vloge uspešno opravljali. Interakcija med člani je ključna za uspešno dokončanje projekta. Ob upoštevanju navedenih dejstev, lahko zaključimo naslednje: čeprav udeleženci ne sledijo predpisanim procesom, lahko vseeno dosežemo boljše rezultate, kot pa če sledimo natančnim predpisom, a med udeleženci ni pravilne komunikacije (Bajec & Krisper, 2003, str. 5).

Drugo načelo zagovarja, da je delujoč program najpomembnejše, kar končni uporabnik dobi, in je to dejstvo pomembnejše kot pa podrobna dokumentacija. Naročnik bo bolj zadovoljen z delujočim programom kot pa s prejemom podrobne in popolne dokumentacije. Dobro pripravljena dokumentacija je pomembna za kasnejše vzdrževanje sistema in lažjo komunikacijo med naročnikom in izvajalcem, tako da je ne smemo popolnoma zanemariti. Dokumentacija mora biti pripravljena tako, da so vsi deli ustrezno utemeljeni (Bajec & Krisper, 2003, str. 5).

Tretje načelo govori o odnosu med izvajalcem in naročnikom, ki je velikokrat preveč formalen, za kar pa ni potrebe, če naročnik in izvajalec dobro sodelujeta. Dobro sodelovanje je ključnega pomena za uspešno vodenje projekta, saj le naročnik ve, kakšen produkt si želi. Dobro sodelovanje med naročnikom in izvajalcem lahko tehnološko zahteven projekt močno poenostavi, na drugi strani pa lahko problemi v odnosih otežijo izvedbo še tako enostavnega projekta (Bajec & Krisper, 2003, str. 5–6).

Zadnje, četrto načelo, navaja, da je odziv na spremembe ključ za uspešnost projekta. Odzivanje na spremembe in sprejem novih zahtev mora metodologija razvoja informacijskega sistema obvladovati skozi celotno fazo projekta. Projektni plani služijo kot okvir, saj je zmotno pričakovati, da se bo v začetnih fazah projekta zajelo vse možne spremembe. Plani morajo tako omogočati tudi spremembe, ki pa ne smejo pretirano

izstopati iz dogovorjenih okvirjev. Sledenje načrtom je pomembno le do stanja, ko se to pretirano ne razlikuje od dejanskega (Bajec & Krisper, 2003, str. 6).

Na podlagi osnovnih štirih načel, ki so jih člani manifesta postavili za temelje agilnih metodologij, so izpeljana priporočila za pomoč pri analizi in gradnji vseh tistih metodologij, ki se uvrščajo med agilne. Ta priporočila so sledeča (Beck in drugi, 2001):

- Zadovoljstvo stranke mora biti na prvem mestu, to pa dosežemo s pravočasnimi dostavami programske opreme.
- Programsko opremo je treba dostavljati pogosto, v časovnih intervalih od nekaj tednov do nekaj mesecev, pri čemer je zaželen čim krajši časovni interval.
- Tudi v času razvoja je treba zagotoviti spremembo zahtev, saj s tem zagotavljamo konkurenčnost stranke.
- Delujoča programska oprema je osnovno merilo za napredek projekta.
- Agilne metodologije zagovarjajo trajnostni razvoj programske opreme, kar pa na drugi strani za razvojno ekipo pomeni konstanten tempo razvoja za nedoločen čas.
- Zaželeno je dnevno sodelovanje med razvojno ekipo in naročnikom.
- Na projektu naj sodelujejo le motivirani posamezniki, katerim je treba zaupati, da bodo delo opravili dobro, hkrati pa jim je treba zagotoviti ustrezno delovno okolje.
- Najučinkovitejša metoda za komunikacijo in posredovanje informacij razvojni ekipi je osebni stik.
- Agilnost metodologij se povečuje z dobro analizo in tehnično odličnostjo.
- Količino nepotrebnega dela zmanjšamo s preprostostjo.
- Samoorganizirane ekipe so sposobne pripraviti najboljše analize, načrte in zahteve za izvedbo projekta.
- Ekipe naj v časovnih presledkih iščejo načine za izboljšanje učinkovitosti, nato pa te ugotovitve implementirajo v delovne naloge.

Najpogostejše agilne metodologije razvoja informacijskega sistema so naslednje (Abrahamsson, Salo, Ronkainen & Warsta, 2002, str. 18–83):

- model Scrum,
- model RUP (ang. Rational Unified Process);
- model XP (ang. Extreme Programming), znan tudi kot ekstremno programiranje,
- model FDD (ang. Feature Driven Development), znan tudi kot funkcionalno voden razvoj informacijskih rešitev,
- model DSDM (ang. Dynamic System Developent Method), znan tudi kot dinamični sistemski model razvoja informacijskih rešitev,
- model ASD (ang. Adaptive Software Development), znan tudi kot prilagodljivi razvoj programskih rešitev,
- modeli družine Crystal.

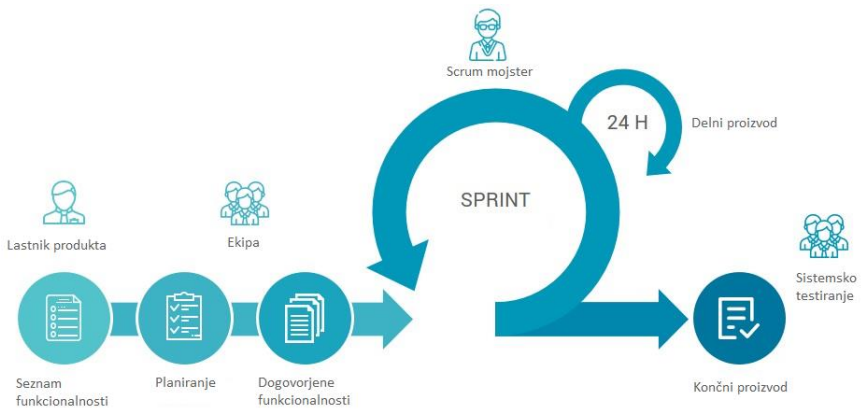
2.3.1 Model Scrum

Scrum je model, ki je osredotočen na delo razvojne skupine. Poudarja, kako naj člani skupine delujejo, da bo pripravljena rešitev dovolj prilagodljiva za delovanje v nenehno spreminjajočem se okolju. Gre za inkrementalni in iterativni model razvoja informacijske rešitve (Awad, 2005, str. 10).

Scrum zagovarja razvoj informacijske rešitve po delih oziroma inkrementih, pri čemer vsak posamezni inkrement potuje skozi ponavljajoče se cikle, kjer se stalno izboljšuje in dopolnjuje. Kot prikazuje Slika 8, je vsak inkrement razdeljen na 30-dnevni cikel, ki ga imenujemo tudi sprint. Na začetku razvoja vsakega sprinta razvojna ekipa oceni delo, ki ga bo treba izvesti v tem ciklu, nato pa se na kratkih dnevnih sestankih poroča o napredku razvoja. Končni rezultat je del testirane in delujoče informacijske rešitve (Abrahamsson, Salo, Ronkainen & Warsta, 2002, str. 27–36).

Podroben opis 30-dnevnega sprinta opisujeta tudi Highsmith (2002, str. 134) in Warcholinski v svojem blogu (2018). V začetni fazi si razvojna ekipa razdeli naloge, nato pa vsak posameznik stremi k uspešni izvedbi naloge. Pri delu se bolj kot na razvojno dokumentacijo in plane zanašajo na pridobljene izkušnje in znanje. O napredku se poroča na rednih dnevnih sestankih.

Slika 8: Model Scrum



Vir: Warcholinski (2018).

Skupne lastnosti podjetij, ki pri razvoju informacijskih rešitev uporabljajo metodo Scrum, so naslednje (Schwaber, 1996, str. 3):

- Velike informacijske rešitve se razdelijo na obvladljive dele tako, da jih v kratkem času lahko razvije majhna skupina razvojnikov.
- Rešitev se oblikuje sistematično ne glede na to, da se na začetku še ne more določiti končnega dizajna rešitve.

- Velike skupine delujejo kot majhne skupinice, saj se delo porazdeli na manjše postopke; skupinice se medsebojno sinhronizirajo, za odpravljanje napak pa se neprestano išče rešitve.

V fazi sprinta se novih zahtev ne sprejema, saj spremembe niso mogoče, kar pa zagotavlja tudi stabilnost projekta (Highsmith, 2002, str. 134). Kljub povečani prožnosti in s tem večjim tveganjem metodologija še vedno zagotavlja ohranjanje nadzora nad projektom, kar ji omogoča sledenje naslednjim načelom, še ugotavlja Schwaber (1996, str. 4):

- Prilagodljivost na spremembe zahtev trga in tehnike zagotavlja implementacijo najboljšega izdelka.
- Zaradi majhnih razvojnih skupin je komunikacija boljša, izmenjuje pa se tudi tiho, neformalno znanje, kar pa vpliva na zmanjšanje režijskih stroškov.
- Delo znotraj posamezne skupine je razdeljeno na posamezne naloge.
- Produkt je vedno na voljo za posredovanje stranki ali konkurenci.

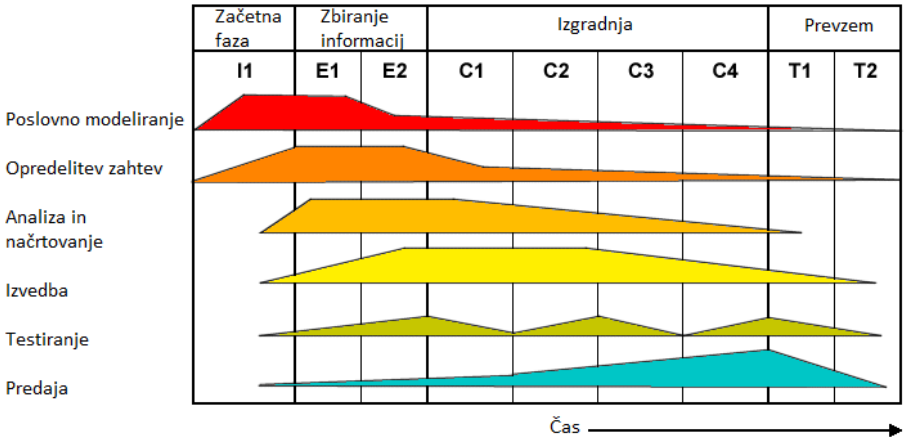
2.3.2 Model RUP

Model RUP so razvili avtorji poenotenega jezika modeliranja (UML – ang. Unifed Modeling Language) z namenom združitve le najboljših postopkov obstoječih procesov razvoja informacijske rešitve. Model ima značilen iterativni pristop, kjer se prav tako kot pri modelu Scrum v vsaki ponovitvi razvije del funkcionalnosti rešitve. Model uporablja vizualno modeliranje s pomočjo jezika UML, obvladuje zahteve, ki so spremljane od začetka do konca razvoja, ter razvija ločene dele informacijske rešitve. Uveden je stalen nadzor nad spremembami ter stalno preverjanje kakovosti in funkcionalnosti rešitve. Na podlagi modela RUP, ki ni model v pravem pomenu besede, saj predstavlja ogrodje, si vsako podjetje prilagodi procese in tehnike tako, da te najbolj ustrezajo razvoju informacijske rešitve. Na Sliki 9 so prikazane faze v razvoju informacijskega sistema, ki si sledijo zaporedno, vendar se zaradi iterativnosti med seboj prepletajo. Vsaka faza je sestavljena iz ključnih aktivnosti (poslovno modeliranje, opredelitev zahtev, analiza in načrtovanje, izvedba, testiranje in predaja naročniku) (Abrahamsson, Salo, Ronkainen & Warsta, 2002, str. 55–61).

Model je sestavljen iz naslednjih faz (Villagomez, 2009, str. 5):

- začetna faza, kjer je cilj pridobiti zanimanje ključnih deležnikov;
- zbiranje informacij, kjer je cilj natančno opredeliti zahteve in arhitekturo sistema;
- izgradnja, kjer je poudarek na razvoju aplikacije do točke, ko je pripravljena za namestitev;
- prevzem, kjer lahko aplikacijo postavimo v produkcijo okolja za uporabo.

Slika 9: Model RUP



Vir: Villagomez (2009).

2.3.3 Model XP

Model XP oziroma metoda ekstremnega programiranja je še ena od metod, ki so se razvile kot odgovor na dolg razvojni cikel tradicionalnih metod življenjskega cikla. Tako je metoda, ki jo prikazuje Slika 10, znana po kratkih razvojnih ciklih, stalnem povratnem toku informacij zaradi dobre komunikacije in inkrementalnem planiranju (Awad, 2005, str. 8).

Model ekstremnega programiranja prinaša hitre rezultate, ker je lahka, prilagodljiva in napovedljiva metodologija razvoja programske opreme. Delujoča rešitev, ki se dostavi naročniku v testiranje, je hitro razvita zaradi povratnega toka informacij in inkrementalnega razvoja. Metodologija spada med agilne, kar pomeni, da se lahko hitro odziva in prilagaja spremembam okolja in naročnika, testi razvijalcev in uporabnikov pa omogočajo hitro odkrivanje ter popravke napak (Beck, 1999).

Osnovni principi ekstremnega programiranja, zaradi katerih ima model prednost pred ostalimi metodologijami, so naslednji (Highsmith, 2002, str. 167–170):

- enostaven načrt,
- test,
- lahkotno planiranje,
- hitre izdaje,
- metafora,
- preoblikovanje,
- programiranje v majhnih skupinah,
- skupno lastništvo programske kode,
- teden s 40 delovnimi urami,

- standard kodiranja,
- integracija.

Spodaj je naštetih pet ključnih načel ekstremnega programiranja, ki jih je treba upoštevati pri realiziranju projekta po metodologiji ekstremnega programiranja (Powell-Morse, 2017).

Načrtovanje:

- potreben popis vseh uporabniških procesov,
- natančen načrt izdajanja verzij,
- projekt je razdeljen na iteracije,
- pogoste majhne izdaje,
- načrtovanje naslednje iteracije začne vsaka končana iteracija.

Upravljanje:

- ekipi je treba pustiti odprte delovne pogoje,
- dan se začne z uvodnim sestankom,
- spremlja se napredek razvoja,
- rotiranje sodelujočih v projektu,
- kadar se ekstremno programiranje zalomi, se ga popravi.

Oblikovanje:

- naj bo preprosto,
- določiti je treba sistemsko metaforo,
- funkcionalnosti ne smejo biti dograjene prezgodaj,
- prestrukturiranje kode kadar koli je to mogoče.

Izgradnja:

- stranka je vedno na voljo,
- koda mora biti zapisana v dogovorjenem jeziku,
- programska koda je programirana v paru,
- en par integrira kode naenkrat,
- vzpostavitev namenskega računalnika za integracijo kode.

Testiranje:

- vsa koda mora imeti enoten test,
- celotna koda mora opraviti vse preizkuse, preden se jo lahko uporabi,
- s testiranjem se odkrijejo tudi napake,
- testi se izvajajo pogosto in rezultati se objavljajo.

Slika 10: Življenjski cikel modela XP

Vir: Powell-Morse (2017).

2.3.4 Model FDD

Gre za funkcionalno voden razvoj informacijskih sistemov in programskih rešitev, kjer je pristop okreten in prilagodljiv ter hkrati ne zajame celotne informacijske rešitve, temveč jo razdeli na manjše funkcionalne celote, ki jih je možno razviti v časovnem roku dveh tednov (Abrahamsson, Salo, Ronkainen & Warsta, 2002, str. 47–55).

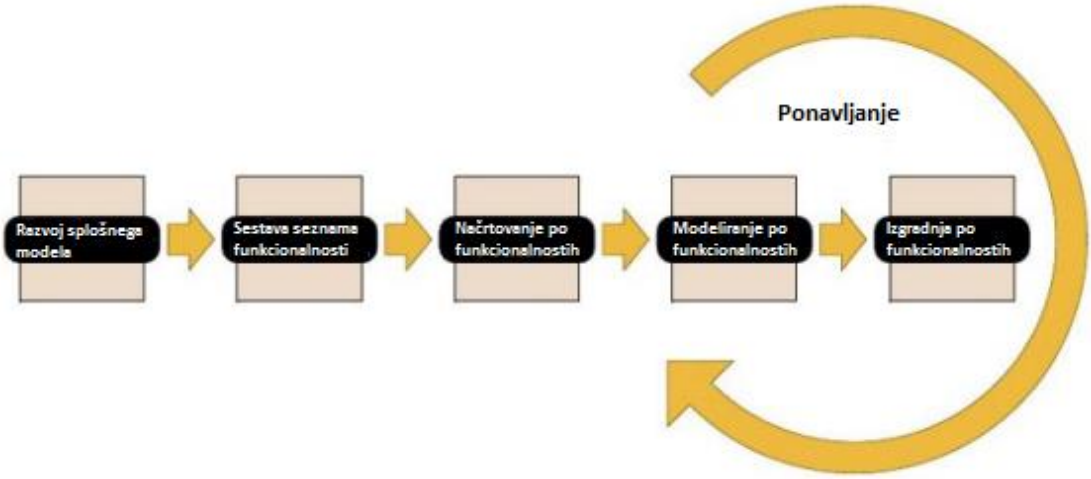
Gre za agilni pristop k razvoju informacijskega sistema, ki se osredotoča na fazo načrtovanja in izgradnje, kar pomeni, da ne pokriva izgradnje celotne informacijske rešitve, čeprav ima možnost uporabe tudi v drugih fazah (Abrahamsson, Salo, Ronkainen & Warsta, 2002, str. 47).

Za uspešno realizacijo projektov je potrebno močno sodelovanje med člani ekipe. Projekt mora biti razdeljen na dovolj majhne funkcionalne celote, da se te lahko razvijejo v kratkem času (Chowdhury, Bhowmik, Hasan & Rahim, 2018)

Model je sestavljen iz petih glavnih aktivnosti, ki se izvajajo iterativno, kar je prikazano na Sliki 11. Prve tri faze so uporabljene na začetku in se jih ne ponavlja, zadnji dve pa sta ponavljajoči se, kar omogoča hitro spremembo informacijske rešitve na podlagi zahtev (Ambler, 2005). Faze modela FDD so naslednje (Awad, 2005, str. 12):

- razvoj osnovnega modela,
- seznam funkcionalnosti,
- načrtovanje po posameznih funkcionalnostih,
- modeliranje funkcionalnosti,
- izgradnja funkcionalnosti.

Slika 11: Model FDD



Vir: *Chowdhury, Bhowmik, Hasan & Rahim (2018).*

2.3.5 Model DSDM

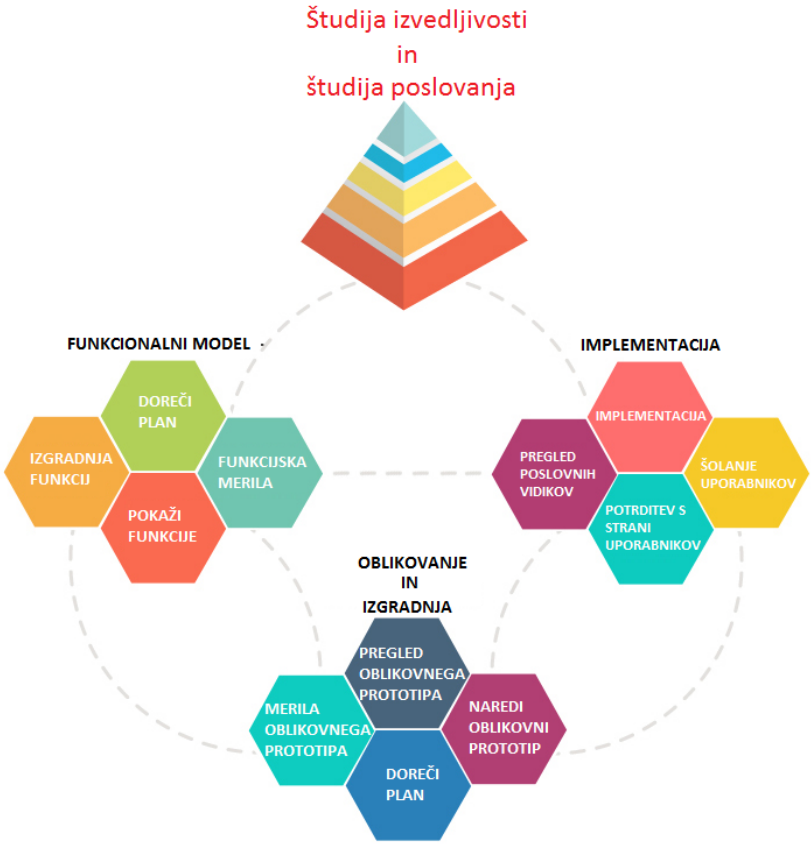
Pri dinamičnem sistemskem modelu razvoja informacijskih rešitev je treba najprej opredeliti čas in sredstva, ki so na razpolago za izgradnjo informacijske rešitve, šele nato začnemo z opredelitvijo funkcionalnosti, ki jih je mogoče razviti v danem okviru časa in z razpoložljivimi sredstvi (Abrahamsson, Salo, Ronkainen & Warsta, 2002, str. 61).

Model DSDM se poleg osredotočenosti na aktivnost ekipe osredotoča tudi na zagotavljanje poslovnih rešitev, kar pa je uresničevanje poslovnega namena projekta, še preden je ta zaključen (Clifton & Dunlap, 2003).

Tako se mora celotna ekipa angažirati in se skupaj odločiti o spremembah. Testiranje in integracija potekata skozi celoten življenjski cikel projekta (Chowdhury, Bhowmik, Hasan & Rahim, 2018, str. 4).

Model je sestavljen iz petih faz, ki so prikazane na Sliki 12. Fazi, imenovani študija izvedljivosti in študija poslovanja, sta zaporedni in se izvedeta natanko enkrat, ostale tri faze, v katerih se izdela funkcionalni model, oblikuje, zgradi in implementira rešitev, pa se ponavljajo v iteracijah vse do zaključka projekta (Abrahamsson, Salo, Ronkainen & Warsta, 2002, str. 61–68).

Slika 12: Model DSDM



Vir: Chowdhury, Bhowmik, Hasan & Rahim (2018).

Alexandrou (2017) navaja, da za model DSDM velja 9 načel, s katerimi se strinjata tudi Clifton in Dunlap (2003), ki dodajata še 10 točko:

- Potrebno je sodelovanje z uporabniki, ki bodo rešitev uporabljali.
- Razvojna ekipa mora imeti pooblastila za sprejemanje odločitev.
- Pogosto je treba dobavljati nove različice programa.
- Kriterij kakovosti je primernost in uporabnost rešitve.
- Realizacija rešitve je zagotovljena z iterativnostjo in inkrementalnostjo.
- Spremembe so reverzibilne.
- Pred dejansko izgradnjo je treba razjasniti najtežje zahteve.
- Testira se skozi celotno izgradnjo.
- V projektu sodelujejo vsi končni uporabniki.
- Pravilo 80 % proti 20 % pomeni, da se 80 % rešitve razvije v 20 % potrebnega časa za celoten razvoj.

2.3.6 Model ASD

Model ASD je znan tudi kot prilagodljivi model razvoja programskih rešitev, kar pomeni, da ima model lastnosti, s katerimi se lahko odziva in prilagaja spremembam v okolju. Tako se model na vsako spremembo odzove s prilagoditvijo in to vključi v razvoj produkta (Highsmith, 2002, str. 173).

Model osrednjo pozornost nameni težavam, ki lahko nastanejo pri razvoju zapletenih rešitev in projektov z veliko spremembami. Tako je delovanje modela prilagojeno nepredvidljivemu okolju, kjer je planiranje projekta oteženo, zato je statični plan razvoja informacijske rešitve zamenjan z dinamičnim razvojnim ciklom (Abrahamsson, Salo, Ronkainen & Warsta, 2002, str. 68–73).

V zapletenem okolju, kjer se striktno držimo načrta, proizvedemo izdelek, ki smo ga nameravali, ampak ta izdelek na koncu ni tak, kakršnega ga potrebujemo. Pri tradicionalnem pristopu so vsakršna odstopanja od načrtov napake, ki jih je treba odpraviti. Pri agilnih metodah pa nas prilagajanja vodijo do izdelkov, ki jih potrebujemo (Highsmith, 2011).

Razvojni cikel modela ASD je razdeljen na tri faze, kar prikazuje Slika 13 (Awad, 2005, str. 14):

- Faza špekulacije predvidi nejasnosti projekta.
- Faza sodelovanja spodbuja timsko delo.
- Faza učenja je namenjena prepoznavanju in odzivanju na napake.

Slika 13: Model ASD

Vir: Highsmith (2011).

2.3.7 Model družine Crystal

Model družine Crystal je sestavljen iz več različnih agilnih metodologij, med katerimi izbiramo na začetku projekta in se tako odločimo za najprimernejšo. Poleg metode

izgradnje model nosi tudi principe za prilagoditev metodologije potrebam projekta (Abrahamsson, Salo, Ronkainen & Warsta, 2002, str. 36).

Model Crystal se je prvič pojavil leta 1998, dokončno pa se je razvil med letoma 2001 in 2004. Model sloni na prepričanju, da je vsak projekt edinstven in da bo za izpolnitev vseh značilnosti projekta morda potreben rahlo prilagojen nabor politik, praks in procesov (Sathram, 2014).

Na Sliki 14 so metodologije označene z barvami, po katerih se imenujejo in ki predstavljajo težavnost. Model nosi dva kriterija, in sicer število sodelujočih in kritičnost. Projekti z več sodelujočimi zahtevajo težjo metodologijo. Kritičnost je označena s črkami (Abrahamsson, Salo, Ronkainen & Warsta, 2002, str. 36–46):

- C (ang. Confort) – izguba udobnosti,
- D (ang. Discretionary) – izguba denarja,
- E (ang. Essenital money) – izguba pomembnega dela denarja,
- L (ang. Life) – izguba življenja.

Slika 14: Model Crystal

		Metodologija Crystal				
		Prazno	Rumeno	Oranžno	Rdeče	Bordo
Kritičnost projekta	(L)	L6	L20	L40	L80	L200
	(E)	E6	E20	E40	E80	E200
	(D)	D6	D20	D40	D80	D200
	(C)	C6	C20	C40	C80	C200
		1 do 6	7 do 20	21 do 40	41 do 80	81 do 200
		Število sodelujočih oseb				

Vir: Sathram (2014).

2.4 Primerjava med tradicionalnimi in agilnimi metodologijami

Kje uporabljati tradicionalno in kje agilno metodologijo ni natančno določeno. V praksi se je že izkazalo, da določene metode le niso primerne za vse projekte (Abrahamsson, Salo, Ronkainen & Warsta, 2002, str. 8).

Ključna razlika med tradicionalnimi in agilnimi metodami je ta, da agilne metode hitro in poceni realizirajo kompleksno rešitev, kljub temu da so zahteve slabo definirane in dokumentacija ni popolna, česar pa tradicionalne metode niso zmožne. Pri agilnih

metodologijah je poudarek na sodelovanju razvojne ekipe, sodelovanju z naročnikom in hitrem odzivanju na spremembe. Tradicionalne pa poudarjajo načrte, dokumentacijo, pogodbe in sledijo predpisanim orodjem za izgradnjo (Yu, Wooi, Wai & Soo, 2012, str. 163).

S tradicionalnimi metodami razvoja se srečujemo že dolgo časa. Pri večini velikih in majhnih projektov se za razvoj uporablja slapovni model, ki se je izkazal za zelo uspešnega. Klub temu pa ima tudi pomanjkljivosti, saj je linearen, kar ne glede na velikost projekta onemogoča fleksibilnost ob spremembah zahtev zaradi obsežne dokumentacije. Kot odgovor na vse to je bila predstavljena metodologija ekstremnega programiranja, ki tudi velja za prvo agilno metodologijo. Z agilnimi metodami lahko uporabljamo več različnih tehnik in se hitro odzivamo na spremembe zahtev. Vsem agilnim metodam sta skupna poudarek na sodelovanju med razvojno ekipo in naročnikom ter dejstvo, da zagovarjajo hitro dobavo programskih rešitev. Razlike med tradicionalnimi in agilnimi metodami prikazuje Tabela 1 (Awad, 2005, str. 35).

Tabela 1: Primerjava tradicionalnih in agilnih metod

	Tradicionalne metode	**Agilne metode**
Pristop	predvidljiv	prilagodljiv
Velikost projekta	veliki	majhni
Merjenje uspeha	po planu	vrednost posla
Možnost sprememb	ni možnih sprememb	spremembe so možne
Način vodenja	centraliziran	decentraliziran
Kultura	nadzor	sodelovanje
Dokumentacija	popolna in veliko	malo
Usmerjenost	na delovni tok	na sodelujoče
Cikličnost	malo	veliko obdobij
Domena	predvidljiva	nepredvidljiva
Načrtovanje vnaprej	podrobno	minimalno
Velikost skupine	velika	majhna
Donosnost naložb	na koncu	na začetku

Vir: Award (2005).

2.4.1 Omejitve tradicionalnih metodologij

Napake pri razvoju informacijskega sistema z uporabo tradicionalnih metodologij po navadi niso posledica slabe ali napačno interpretirane metodologije, ampak nastanejo zaradi pomanjkanja poznavanja okolja naročnika, v katerem deluje, ali napak pri analiziranju in modeliranju poslovnega okolja (Zornada, 2002, str. 223).

Tradicionalne metodologije so znane po svoji uspešnosti, manj znane pa so njihove slabosti. Največja kritika na račun tradicionalnih metod je, da so zelo birokratske, saj

predpisujejo vodenje velike količine spremne dokumentacije projekta, kar privede do dejstva, da se zaradi nje upočasni tempo razvoja. Druga kritika je, da tradicionalne metodologije temeljijo na zaporednem razvoju, kar pomeni, da sta testiranje in s tem povratne informacije stranke postavljena v zadnjo fazo življenjskega cikla projekta. Pomanjkljivost projekta se tako ugotovi šele na koncu, ko bi ta moral biti že zaključen, kar pa je lahko tudi kritično (Fowler, 2005).

Lastnost tradicionalnih metodologij je, da je treba že pred začetkom izgradnje definirati vse zahteve in spremembe, kar pa je potencialna težava, saj se zahteve lahko spreminjajo in zaradi hude konkurence to predstavlja omejitev teh tehnologij. Ostale omejitve so še (Gradišar, Jaklič, Damij & Baloh, 2005, str. 272–273):

- slabo upoštevane potrebe managerjev, saj jim sistem ne nudi zadostne podpore pri odločanju in ne vključuje strateških in taktičnih ciljev;
- razvoj velikih projektov, ki lahko traja več let, kar pa lahko privede do tega, da ob implementaciji sistem ne bo zadovoljeval uporabnikovih zahtev in potreb;
- dokumentacija projekta, ki je namenjena izključno osebam, ki sodelujejo pri analizi in izgradnji, ne pa tudi končnim uporabnikom;
- neambiciozno načrtovanje, ki lahko privede do tega, da se avtomatizira le delo, ki se je prej opravljalo ročno, kar pa pomeni, da sistem ne prinese radikalnih sprememb in dejanske koristi;
- zaradi dolgotrajnega razvoja se lahko zgodi, da se nujne zahteve implementira v obstoječi sistem brez ustreznih načrtov, kar pa privede do neobvladljivega sistema.

Uspeh projekta je odvisen od sposobnosti odzivanja metodologije na spremembe zahtev naročnika in trga. Prav sprejemanje zahtev predstavlja ključno razliko med tradicionalnimi in agilnimi metodologijami (Williams & Cockburn, 2003, str. 39–43). Trditev poudarjajo tudi Gradišar, Jaklič, Damij in Baloh (2005, str. 269), ki navajajo, da so visoka cena, dolgotrajnost in omejene možnosti za spremembe glavne omejitve tradicionalnih metodologij razvoja informacijskega sistema. Zato so te bolj primerne za rešitve, kjer so zahteve sprememb zelo dobro definirane.

Tudi Fowler (2005) v svojem delu navaja, da je nezmožnost upoštevanja sprememb ključna omejitev tradicionalnih metodologij – v daljšem življenjskem ciklu namreč ne dovoljujejo sprememb funkcionalnosti projekta, v današnjem okolju, kjer se tehnologija in zahteve zelo hitro spreminjajo, pa je projekte skoraj nemogoče izvesti na tak način.

Tradicionalne metodologije so usmerjene na postopke, procese in dokumentacijo, ne pa na ljudi in delo z njimi. Cilj je usmerjen k pripravi rešitve, ki bo ne glede na njihovo znanje in izkušnje primerna za vse uporabnike. Dobro predpisan razvojni proces v realnosti ne more nadomestiti pomanjkanja znanja in izkušenj razvojne ekipe, obratno pa lahko izkušnje nadomestijo slabo definirane procese (Fowler, 2005). Podobnega mnenja sta tudi Cockburn

in Highsmith (2001, str. 131–133), ki še dodajata, da so ljudje najpomembnejši dejavnik pri razvoju programske opreme.

2.4.2 Omejitve agilnih metodologij

Kljub temu da agilne metodologije prinašajo veliko prednosti v primerjavi s tradicionalnimi, pa imajo tudi slabosti, ki so se pojavile pri njihovi praktični uporabi. Ena ključnih slabosti je znatno zmanjšanje in celo zavračanje dokumentacije, saj v ospredje postavljajo programsko kodo in v njej evidentirane opombe razvijalcev (Yu, Wooi, Wai & Soo, 2012, str. 162).

Tradicionalne metodologije zmanjšujejo tveganje projekta tako, da vlagajo v arhitekturo rešitve, kar pa zunanjim strokovnjakom olajša uporabo in vpogled v strukturo programske rešitve, čeprav to podražuje celoten projekt. Agilne metodologije zaradi zapostavljene dokumentacije iščejo oporo v izkušnjah in komunikaciji med člani tima. Vse skupaj sloni na delitvi tihega znanja med člani tima, saj to znanje ni zapisano v dokumentacijo, kar pa lahko vodi do napak pri arhitekturi rešitve, ki pa jih zunanji pregledovalci zaradi slabe dokumentacije ne morejo odkriti (Boehm, 2002, str. 64–69).

Fowler (2005) pojasnjuje, da komunikacija na štiri oči postaja vse težja, ko na enem projektu sodeluje več kot 20 razvijalcev, in začne predstavljati celo oviro, čeprav je to značilnost agilnih metodologij. Ravno sodelovanje večjih skupin predstavlja glavno omejitev agilnih metodologij, kar ugotavljata tudi Cockburn in Highsmith (2001, str. 131–133), ki razlagata, da je agilni razvoj za večje skupine težji kot za manjše.

Glavno načelo agilnih metodologij pravi, da je največja prednost zadovoljiti kupca s hitro in redno dostavo delujoče programske rešitve, kar pa ni vedno prednost in lahko postane tudi ovira. Zgodaj dostavljene programske rešitve lahko po testiranju vsebujejo veliko napak, kar pa pomeni, da bo potrebna večja predelava programske rešitve. Večina agilnih metodologij ne zadosti inšpekcijskim pregledom programske kode v življenjskem ciklu projekta, ampak samo zaupa ocenam kontrolnih mehanizmov in jim sledi (Boehm, 2002, str. 63–69).

Če so ljudje, ki sodelujejo na projektu, dovolj dobri, lahko z uporabo katerega koli postopka vsako nalogo dobro izpolnijo. Če niso dovolj dobri, tudi postopek, ki ga uporabimo, ne bo prikril njihove neustreznosti. Cockburn in Highsmith (2001, str. 131–133) zato uspeh projekta pripisujeta ljudem. Tudi Awad (Awad, 2005) potrjuje to dejstvo in dodaja, da je uspeh agilnih metodologij rezultat ekipe izkušenih ljudi in ne načela metodologije.

Nujni del ekipe je tudi naročnik, saj je vključen v procese razvoja programske opreme in s svojim znanjem pomaga pri razvoju in testiranju verzij. Problem lahko nastane, če naročnik nima zadostnega znanja za potrebe celotnega razvojnega obdobja. Dodaten

problem lahko nastane, če naročniki niso usklajeni oziroma med njimi prihaja do trenj (Boehm, 2002, str. 64–69).

3 ANALIZA INFORMACIJSKE REŠITVE

3.1 Kratka predstavitev izbranega zavoda

Izbrani zavod skrbi za ohranjenost in sonaravni razvoj slovenskih gozdov in vseh njihovih funkcij za kakovostno gospodarjenje. Zavod črpa sredstva iz proračuna, saj je neposredni proračunski uporabnik, deluje pa v skupini uporabnikov javnih zavodov ali drugih izvajalcev javnih služb.

Znotraj zavoda je izoblikovanih več oddelkov. Ti se delijo na oddelek za gozdnogospodarsko načrtovanje, oddelek za gojenje in varstvo gozdov, oddelek za gozdno tehniko in razvoj podeželja, oddelek za gozdne živali in lovstvo, oddelek za kadrovsko pravne zadeve, oddelek za informatiko in oddelek za finančne zadeve. Zadnji oddelek se ukvarja tudi z naročanjem materiala preko naročilnic, ki je hkrati tudi proces, ki ga želijo informatizirati. Zavod ima sedež v centralni enoti v Ljubljani. Na regionalni ravni so prisotni v štirinajstih območnih enotah, sledi pa jim še 69 krajevnih enot in 396 gozdnih revirjev. V sestavi imajo tudi 10 lovišč (Izbrani zavod, 2018).

Izbrano podjetje že dlje časa uporablja finančno aplikacijo, ki jo je razvilo podjetje za računalniški inženiring. V tej finančni aplikaciji je programska rešitev za področja računovodstva, financ, osnovnih sredstev in analiz. Sistem je prilagodljiv, povezljiv in razširljiv ter je zavodu v pomoč pri ustreznem prilagajanju na vsakodnevne spremembe v zunanjem in notranjem okolju (Podjetje za računalniški inženiring, 2008).

Zaradi uspešne uporabe finančne aplikacije in njenih modulov (eRačuni, ePoslovanje) so se na podjetje za računalniški inženiring obrnili z željo po nadgradnji modula ePoslovanja, v katerem bi vodili tudi eNaročilnice. Zaradi svoje modularne zasnove ePoslovanje omogoča prilagodljivo delovno okolje, saj lahko uporabnik upravlja tudi s posameznim delom znotraj modula glede na vlogo, ki mu je dodeljena. Trenutno uporabljajo ePoslovanje za finance in DDV, možnosti uporabe pa so neomejene (Podjetje za računalniški inženiring, 2008).

Zaradi razvejanosti uporabnikov po vsej Sloveniji je bilo treba narediti natančen popis trenutnega delovnega toka ter temu prilagoditi nov delovni tok, ki bo kar se da optimiziral in poenostavil postopek naročanja materiala. V obstoječem delovnem toku so podatke vnašali na fizične dokumente, vsako naročilnico pa so spremljali še tiskani dokumenti pridobljenih ponudb za njihovo lastno evidenco. Pred analizo so bili organizirani sestanki, kjer so bili opravljeni razgovori s ključnimi zaposlenimi, ki najbolje poznajo trenutni delovni tok ter vodjo informatike. Na podlagi izbranih dejstev se je izdelal natančen popis

obstoječega delovnega toka za naročanje v izbranem zavodu, ki je naslednji (Podjetje za računalniški inženiring, 2017):

- Pripravljavec vnese novo naročilnico.
  V prvi fazi se vnese območno enoto, kjer se naročila vrši, nato vrsto naročila (blago, storitve, gradnje) za njihovo lastno evidenco, stroškovno mesto za knjiženje ter ponudnika, ki morata biti vsaj dva, in predmet naročila. Številčenje dokumenta je avtomatsko in na nivoju izbranega zavoda, prav tako pa se tudi avtomatsko predlaga sistemski datum.
- Uradni predlagatelj naročilnico potrdi ali zavrne.
  Uradni predlagatelj navadno preveri naročilnico in jo potrdi oziroma jo lahko pripravljavcu zavrne. Na nekaterih območnih enotah je lahko pripravljavec tudi uradni predlagatelj, zato lahko v izogib pošiljanju naročilnice samemu sebi tudi predlagatelj vnese naročilnico. V tem primeru izpolni vse, kar je opisano v prejšnji vlogi.
- Knjigovodja dopolni podatke.
  Knjigovodja območne enote vnese stroškovno mesto in konto ter posreduje naročilnico v podpis vodji organizacijske enote.
- Vodja organizacijske enote naročilnico potrdi ali zavrne.
  Vodja organizacijske enote potrdi naročilnico oziroma naročilnico zavrne z obvezno pripombo knjigovodji.
- Tajnica naročilnice preveri če naročilnica potrebuje dodatno odobritev.
  Tajnica naročilnice račun knjiži oziroma ga pošlje v dodatno potrditev vodji finančno računovodske službe.
- Vodja finančno računovodske službe dodatno potrdi naročilnico.
  Vodja finančno računovodske službe v delovnem toku naročilnice to dodatno potrdi in naročilnica prispe nazaj k tajnici, ki nato zaključi naročilo. Vodja lahko naročilnico tudi zavrne in v tem primeru se vrne nazaj do uradnega predlagatelja.

Za potrebe eNaročilnic se je tako izdelala informacijska rešitev za aplikacijo ePoslovanje.

3.2 Metodologija razvoja informacijske rešitve v podjetju

Predpisana metodologija razvoja aplikacij v podjetju za računalniški inženiring je osnovana na metodologiji Oracle CASE, vendar se lahko po potrebi prilagaja zahtevam posameznega projekta oziroma aplikacije. Za vsako prilagajanje metodologije je odgovoren projektant, ki skrbi za projekt, vsako spremembo pa mora odobriti tudi vodja projekta. Vsaka sprememba, predvsem pa odstopanje od predpisanih točk metodologije, mora biti preverjena in odobrena tudi s strani vodje kakovosti ter za člane projektnega tima tudi dokumentirana v internem navodilu in vzpostavitvenem dokumentu projekta. Obseg dela je odvisen od natančnosti naročnikovih zahtev in njihove interne dokumentacije. V nekaterih primerih je treba najprej izdelati posnetek stanja naročnika, v drugih primerih pa

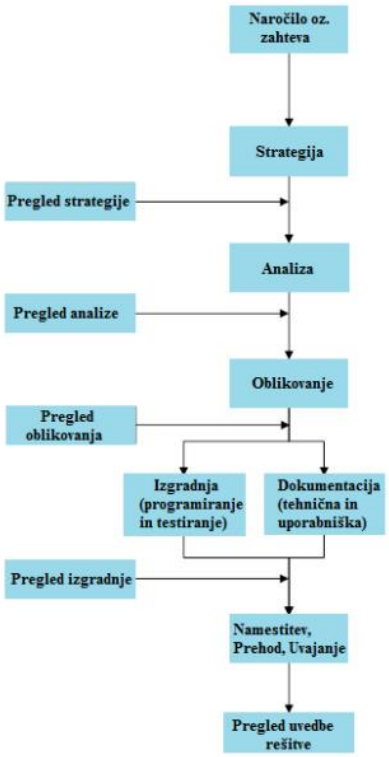
specifikacije naročnika zadostujejo kriterijem faze analize (Podjetje za računalniški inženiring, 2014).

Slika 15 prikazuje osnovno shemo razvoja rešitev v podjetju za računalništvo in informatiko. V postopek se lahko naknadno dodajo tudi dodatne aktivnosti, ki so odvisne od zahtevnosti projekta ter dogovora z naročnikom. Tako se v veliko primerih na koncu doda še izobraževanje končnih uporabnikov (Podjetje za računalniški inženiring, 2014).

Analiza je ključni del vsakega projekta ne glede na to, ali gre za dopolnitev ali za izgradnjo nove programske rešitve, zato mora natančno definirati vse postopke, da pri izgradnji ne bo prihajalo do nesporazumov in da bo končna rešitev ustrezala potrebam naročnika. Vsako analizo preverita tako izvajalec kot tudi naročnik, ki dodatno celotno analizo tudi potrdi. Po potrditvi se začne izgradnja, ki strogo temelji na analizi. Če se tekom izgradnje pojavijo dodatna vprašanja, se izvajalec in naročnik posvetujeta ter po potrebi dopolnita analizo, saj je dokumentacija ključna tudi zaradi uveljavljanja garancije naročnika (Podjetje za računalniški inženiring, 2010).

Dokument analize je namenjen natančnemu opisu poslovnih procesov in funkcionalnosti želja naročnika. Vsi koraki v procesu morajo biti natančno definirani in opisani. Analiza temelji na prejeti specifikaciji naročnika, posnetku stanja ali zakonskih določilih (Podjetje za računalniški inženiring, 2011).

Slika 15: Shema procesa razvoja



Vir: Podjetje za računalniški inženiring (2014).

Za življenjski cikel razvoja se uporablja metodologijo Oracle CASE, ki predstavlja množico navodil in tehnik navzkrižnega preverjanja. Z njimi se vsako točko razvoja lahko preveri, da se ugotovi ustreznost rezultatov in izpolnjevanje pogojev, predpisanih v analizi. Metoda razvoja je klasični slapovni model razvoja, pri katerem se vsaka naslednja faza začne s končanjem prejšne in med fazami ni prekrivanja. Ključni vzrok za uporabo takšnega modela je sodelovanje podjetja za računalniški inženiring pri razvoju programskih rešitev s proračunskimi uporabniki, ki zahtevajo takšen način razvoja programskih rešitev.

Za izbrano podjetje je bil v začetku izbran slapovni model, saj je šlo za razvoj po korakih, od analize do končne rešitve produkta. V fazi testiranja produkta pri naročniku je bilo ugotovljeno, da rešitev, ki je bila analizirana in potrjena z njihove strani, ne ustreza njihovemu načinu dela. Med popisom delovnega toka namreč niso vključili vseh udeležencev, kar je povzročilo, da niso vključene vse posebnosti in tako se delo ni poenostavilo, ampak se je le še otežilo. Na podlagi dostavljene verzije je bila izvedena nova analiza, v kateri so bile upoštevane vse njihove specifične želje in interni predpisi, ter ponovno ocenjena količina dela, potrebnega za spremembo programske rešitve. Naročnik je dobro preveril analizo ter jo potrdil podjetju za računalniški inženiring. V drugi fazi se je zaradi specifičnosti uporabila metodologija poenotenega procesa. To je objektno usmerjena metodologija, ketere temelj so inkrementi in iterativni razvoj. Projekt je bil razdeljen na več manjših projekov, pri čemer je vsak predstavljal del aplikacije, na koncu pa so bili združeni. Razvoj aplikacije je temeljil na orodju Oracle CASE. Znotraj iteracij je bil uporabljen klasičen slapovni model. Kontrola iteracij je temeljila na predpisih metodologije Oracle CASE, pri čemer je bil kljub agilnemu pristopu mogoč dober nadzor nad projektom.

3.3 Analiza aplikacije eNaročilnice

Za namen vnašanja in obdelave naročilnic bo v okviru obstoječe aplikacije ePoslovanje izdelan nov krogotok z vlogami in maskami za evidentiranje eNaročilnic. Aplikacija bo komunicirala tudi z drugimi moduli finančne aplikacije zaradi vseh potrebnih šifrantov.

Aplikacija ePoslovanje je sodobna spletna rešitev, ki z ustreznimi nastavitvami omogoča podporo različnim delovnim tokovom v zavodu. Aplikacija je lahko samostojna rešitev ali rešitev, ki je povezana s strani naročnika izbranim poslovnim informacijskim sistemom. Ob implementaciji se tako lahko aplikacija ePoslovanje poveže s poljubno informacijsko rešitvijo in poljubnim arhivskim sistemom oziroma lahko deluje povsem samostojno.

Ker gre za spletno aplikacijo, je ta narejena v tehnologiji Microsoft ASP.NET, deluje pa na Microsoftovem spletnem strežniku IIS. Ogrodje za izgradnjo je .NET Framework 4.5, v izbranem zavodu pa uporablja podatke finančne aplikacije, ki je shranjena v bazi Oracle 12c. Aplikacija do vseh potrebnih podatkov dostopa preko rešitve NHibernate, ki je

odprtokodni objektno-relacijski vmesnik in se uporablja za preslikavo med podatki v bazi in objektnimi modeli v aplikaciji.

V standardni različici je aplikacija ePoslovanje povezana z informacijskimi rešitvami podjetja za računalniški inženiring, in sicer s Finančno aplikacijo. Analiza rešitve eNaročilnice temelji na novem delovnem toku, hkrati pa omogoča tudi sprotno vizualizacijo naročilnice ob vnašanju in shranjevanju podatkov.

3.3.1 Funkcionalnosti nove aplikacije eNaročilnice

Nova aplikacija eNaročilnice bo v ePoslovanju uporabnikom omogočala:

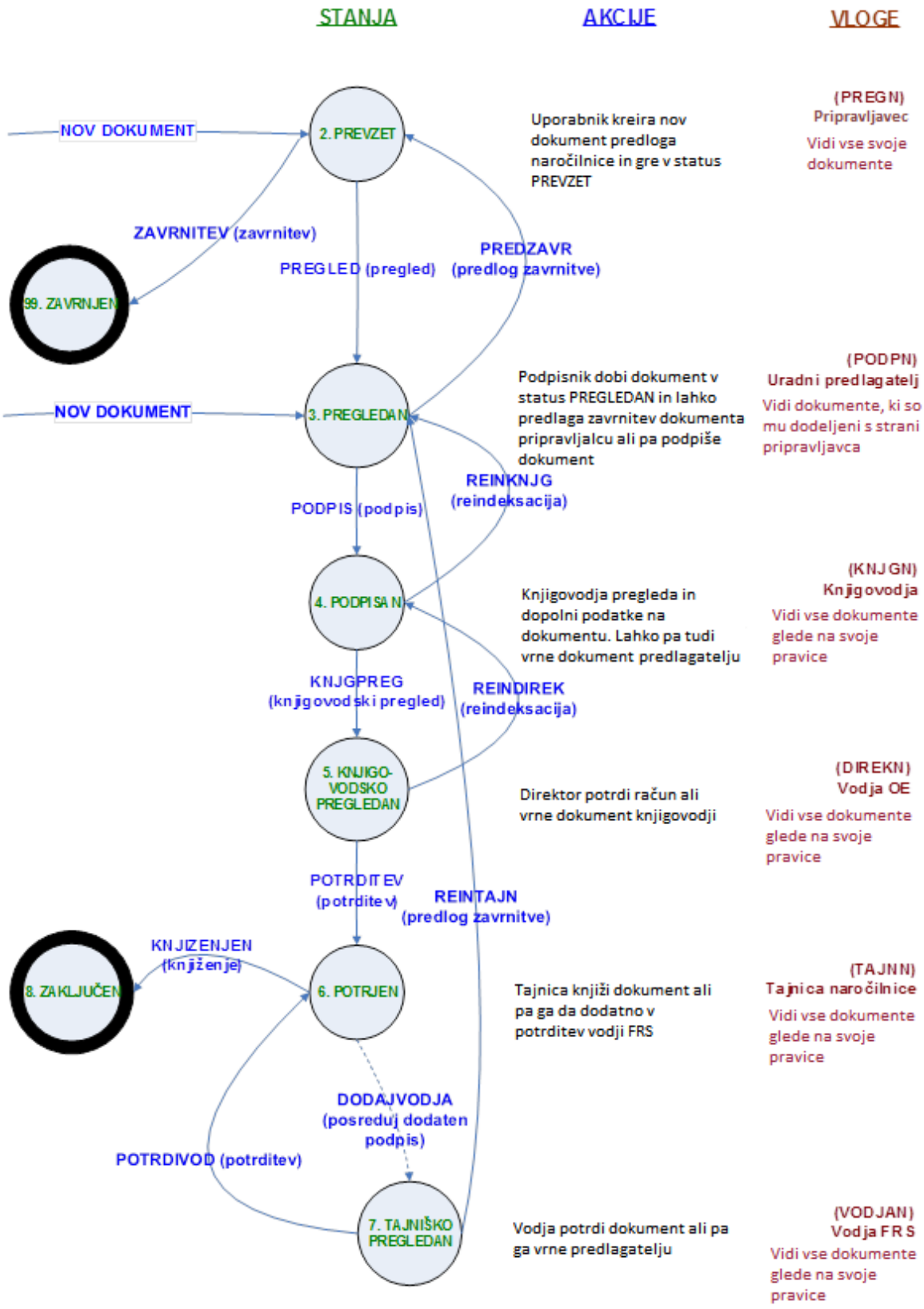
- vnašanje novih eNaročilnic ter spremljanje statusov in komentarjev skozi celotni krogotok;
- pripenjanje pridobljenih ponudb k naročilnicam v pdf-formatu, ki so obvezni del navodil po interni specifikaciji;
- samodejno polnjenje nekaterih atributov (številka naročilnice, sistemski datum, območna enota, vrsta dokumenta);
- obdelavo eNaročilnic po predhodno nastavljenem delovnem toku;
- urejanje vlog uporabnikov in njihovih pravic za delo z eNaročilnicami;
- tiskanje vizualizirane naročilnice za lastno evidenco ter tiskanje eNaročilnice za pošiljanje k partnerju;
- izpise iz aplikacije na podlagi različnih s strani uporabnika vnesenih kriterijev iskanja;
- nadomeščanje odstotnih uporabnikov s strani pooblaščenega namestnika, da obdelava naročilnic nemoteno poteka naprej;
- beleženje vseh sprememb v za to namenjenih dnevnike, zaradi zagotavljanja revizijske sledi;
- obveščanje uporabnikov o čakajočih dokumentih neposredno na njihov elektronski naslov po predhodno nastavljenem intervalu pošiljanja.

3.3.2 Delovni tok eNaročilnice

eNaročilnice temeljijo na samostojnem delovnem toku, v katerem sodelujejo različni uporabniki, ki prihajajo iz vseh 14 območnih enot, dodeljene pa imajo tudi različne vloge. Vsaka vloga nosi določene pravice in zadolžitve, za katere je posameznik odgovoren v fazi, v kateri se eNaročilnica takrat nahaja.

Kje v delovnem toku se eNaročilnica nahaja, je odvisno od stanja dokumenta, saj so vloge povezane s stanji, tako da določena vloga lahko obdeluje vnaprej določeno stanje dokumenta. Vsako stanje ima nastavljeno predhodno in naslednje stanje. Slika 16 prikazuje delovni tok eNaročilnice, v katerem so natančno definirane vloge, prehodi stanj in akcije, ki so uporabniku na voljo v določenem stanju.

Slika 16: Delovni tok eNaročilnic



Vir: Podjetje za računalniški inženiring (2018).

Z vlogo uporabnikov so povezane tudi aktivnosti posameznih menijev, prikazov in gumbov. Uporabnik ima lahko več kot eno vlogo. Administrator aplikacije ima največ pravic in lahko v primeru težav ureja tudi dokumente, ne sodeluje pa v delovnem toku. Tako lahko spreminja podatke in status dokumenta, če je le tega potrebno vrniti k pravemu uporabniku.

Poleg vloge, ki se določi uporabniku, pa je treba določiti tudi pravice za delo v območni enoti, kjer uporabnik deluje. Lahko se določi tudi celotno organizacijo – takšno pravico bo imel vodja FRS, saj v potrditev dobiva eNaročilnice iz vseh območnih enot. Tako kombinacija vloge in pravice za delo v območni enoti uporabniku v aplikaciji ePoslovanje omogočata vnašanje in obdelavo eNaročilnic izključno v okviru območne enote, za katero je zadolžen. Takšna ureditev je uporabniku bolj prijazna, saj vsak uporabnik vidi samo svoje dokumente ter na izpisih pregleduje dokumente območne enote, za katero je zadolžen.

Opis stanj eNaročilnic ter akcij, ki jih uporabnik lahko izvede:

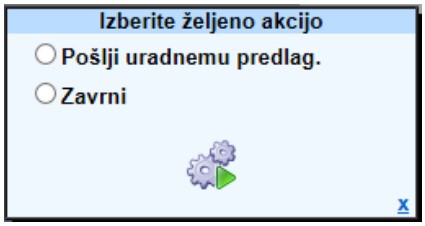
- Prevzet. To je prvo stanje in pomeni, da je uporabnik začel z vnašanjem nove eNaročilnice. Po končanem vnosu ima uporabnik na voljo naslednje akcije:
    - Pošlji uradnemu predlagatelju
    - Zavrni
- Pregledan. Gre za drugo stanje in uporabniku so na voljo naslednje akcije:
    - Potrdi predlog
    - Predlog zavrnitve
- Podpisan. Tukaj imajo uporabniki na voljo akcije:
    - Pošlji direktorju
    - Zavrni pripravljavcu
- Knjigovodsko pregledan. V tem stanju so možne akcije:
    - Potrdi tajnici
    - Zavrni knjigovodji
- Potrjen. To je predzadnje stanje, kjer uporabnik izbira med naslednjima akcijama:
    - Knjiženje
    - Posreduj dodaten podpis
- Tajniško pregledan. Gre za zadnje aktivno stanje, kjer sta uporabniku na voljo:
    - Potrditev
    - Predlog zavrnitve
- Zaključen. To je zadnje stanje – tu urejanje dokumenta ni več mogoče, mogoče pa je tiskati spremno dokumentacijo ter vizualizirano naročilnico in eNaročilnico.
- Zavrnjen. Gre za še eno dokončno stanje, kjer prav tako ni več mogoče urejati eNaročilnice.

Ker so stanja dokumentov in vloge povezane, je spodaj opis vseh vlog ter akcij, ki jih ima določena vloga. Uporabniki imajo lahko naslednje vloge:

- PREGN Pripravljavec naročilnice
- PODPN Uradni predlagatelj naročilnice
- KNJGN Knjigovodja naročilnice
- DIREKN Vodja OE naročilnice
- TAJNN Tajnica naročilnice
- VODJAN Vodja FRS naročilnice

Pripravljavec naročilnice ima dodeljeno vlogo za vnos novih naročilnic. Ko vnese vse potrebne podatke vključno s pridobljenimi ponudbami, ima, kot je prikazano na Sliki 17, na voljo dve akciji.

Slika 17: Akciji pripravljavca

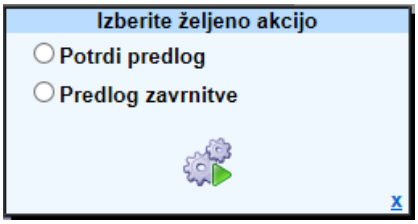


Vir: Podjetje za računalniški inženiring (2018).

- »Pošlji uradnemu predlagatelju« pomeni, da bo eNaročilnica poslana v nadaljnjo obdelavo k uradnemu predlagatelju. Obvezno morajo biti izpolnjeni zahtevani atributi, v nasprotnem primeru aplikacija ne dovoli nadaljevati in uporabniku javi opozorilo o tem, kateri atribut mora izpolniti za uspšeno pošiljanje v nadaljnjo obdelavo.
- »Zavrnitev« pomeni, da se eNaročilnica zavrne in je ni več mogoče urejati.

Uradni predlagatelj naročilnice je drugi v nizu obdelave eNaročilnic. Naročilnico pregleda in vsebinsko dopolni ter vnese zahtevane atribute za to vlogo. V nakaterih območnih enotah tudi uradni predlagatelj odpre novo eNaročilnico, zato ima tudi ta vloga možnost dodajanja novih eNaročilnic. Po končanem vnosu ima na voljo dve akciji, kar prikazuje Slika 18.

Slika 18: Akciji uradnega predlagatelja

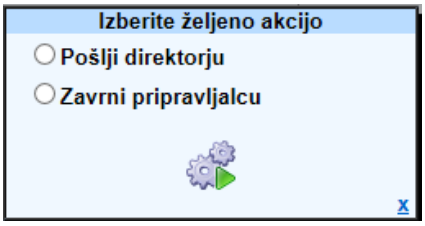


Vir: Podjetje za računalniški inženiring (2018).

- »Potrdi predlog« pomeni, da se uradni predlagatelj strinja z naročilom in pošlje eNaročilnico v knjigovodsko dopolnitev h knjigovodji.
- »Predlog zavrnitve« eNaročilnico vrne pripravljavcu, pri tem pa mora uradni predlagatelj vnesti tudi razlog zavrnitve.

Knjigovodja naročilnice vnese knjigovodske podatke, kot so konto, stroškovni nosilec in stroškovno mesto. Slika 19 prikazuje akciji, ki sta mu na voljo.

Slika 19: Akciji knjigovodje

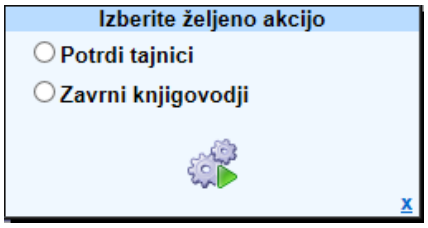


Vir: Podjetje za računalniški inženiring (2018).

- »Pošlji direktorju« pomeni, da knjigovodja po vpisanih zahtevanih atributih pošlje eNaročilnico na najvišji nivo dotične območne enote, torej vodji območne enote.
- Z ukazom »Zavrni pripravljavcu« se naročilnica vrne pripravljavcu naročilnice, pri čemer je nujno treba vpisati tudi razlog zavrnitve.

Vodja OE naročilnice preveri vse podatke, saj ima vsa polja osivena in ne more ničesar več urejati. Vodji sta na voljo 2 akciji, ki sta prikazani na Sliki 20.

Slika 20: Akciji vodje območne enote

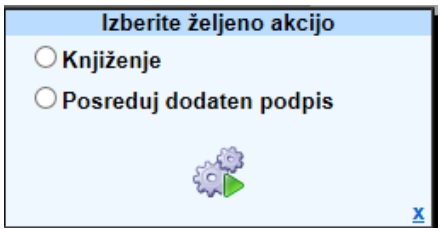


Vir: Podjetje za računalniški inženiring (2018).

- »Potrdi tajnici« pomeni, da se vodja OE strinja z naročilom in tako tajnica dobi eNaročilnico.
- Ukaz »Zavrni knjigovodji« pa vrne eNaročilnico knjigovodji. Potreben je vnos razloga zavrnitve, v nasprotnem primeru aplikacija ne dovoli nadaljevanja.

Tajnica naročilnice je v vseh primerih zadnja vloga. Določene eNaročilnice mora še dodatno posredovati v podpis, ampak se po podpisu vodje finančno računovodske službe vrnejo nazaj k njej. Kot prikazuje Slika 21, ima tajnica na voljo spodnji akciji.

Slika 21: Akciji tajnice

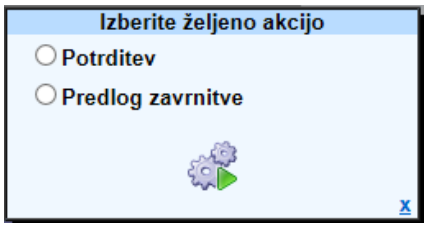


Vir: Podjetje za računalniški inženiring (2018).

- »Knjiženje« zaključi eNaročilnico ter avtomatsko pripravi eNaročilnico v pdf-formatu.
- Ukaz »Posreduj dodatnemu podpisniku« pa naročilnico pošlje še v dodaten podpis k vodji finančno računovodske službe.

Vodja FRS naročilnice je tako zadnja vloga, ki potrjuje ustreznost eNaročilnice. Vodja preveri ustreznost zneska in izbere eno izmed dveh akcij, ki ju ima na voljo. Akciji sta prikazani na Sliki 22.

Slika 22: Akciji vodje FRS



Vir: Podjetje za računalniški inženiring (2018).

- Ukaz »Potrditev« pomeni, da potrjuje eNaročilnico in le-to dobi tajnica, ki nato zaključi proces.
- »Predlog zavrnitve« pa vrne eNaročilnico uradnemu predlagatelju, nujno potreben pa je še vnos razloga zavrnitve.

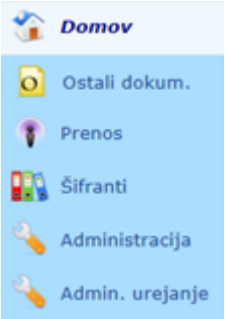
3.3.3 Opis menijev

Pred prvo uporabo je treba v aplikacijo vnesti uporabnike. To uredi administrator v šifrantu uporabnikov. Uporaba programa je vezana na operacijski sistem, tako da je treba v šifrant vnesti ime uporabnika za avtentikacijo v operacijskem sistemu Windows. Vsakemu uporabniku se določi tudi vloge, za katere je zadolžen, ter delovni tok v procesu. Z uspešno prijavo se vedno prikaže meni na zavihku »Domov«, ki služi kot pozdravno okno z vsemi informacijami o čakajočih eNaročilnicah.

Slika 23 prikazuje meni na levi strani, ki je uporabniku vedno na voljo, odvisen pa je od uporabnikove vloge. Zajema sledeče menije:

- Domov,
- Ostali dokumenti,
- Prenos,
- Šifranti,
- Administracija,
- Administrativno urejanje.

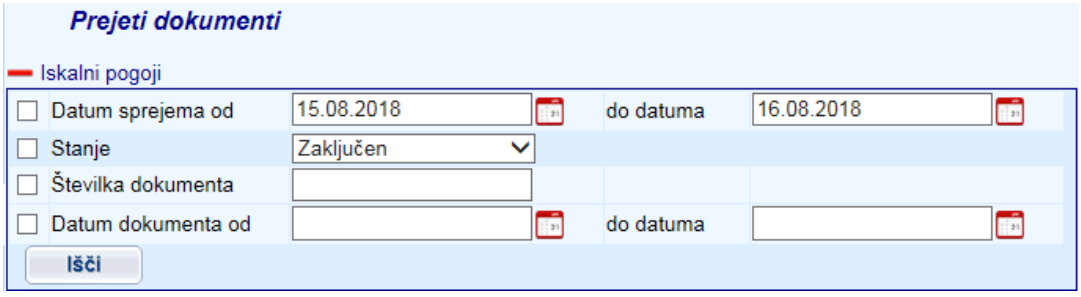
Slika 23: Meniji uporabnika



Vir: Podjetje za računalniški inženiring (2018).

Meni »Ostali dokumenti« je namenjen pregledovanju eNaročilnic, ki so še v obdelavi. Uporabnik bo lahko iskal le med eNaročilnicami, ki so v njegovi območni enoti. Meni je uporabniku prijazen, ker sestoji iz ključnih polj, po katerih lahko uporabnik išče. Po vstopu v meni se odpre forma z iskalnimi pogoji, kjer s kljukico pred poljem izberemo vrsto filtra, po katerem želimo iskati. Slika 24 prikazuje polja, po katerih je moč iskati.

Slika 24: Prejeti dokumenti



Vir: Podjetje za računalniški inženiring (2018).

Meni »Prenos« trenutno še nima funkcionalnosti. Namenjen bo tajnici naročilnice. V naslednji fazi, ko bodo eNaročilnice že v uporabi, se bo analiziralo in vzpostavilo arhiviranje dokumentacije v oblaku pri izbranem ponudniku.

Meni »Šifranti« je namenjen pregledu šifrantov, ki so potrebni za nemoteno delovanje modula. Večine šifrantov se tukaj ne bo dalo urejati, saj služijo le kot prikaz iz finančnega modula. Takšni šifranti so recimo organizacijske enote, poslovni partnerji, kontni plan.

Ima pa ePoslovanje dva specifična šifranta, in sicer:

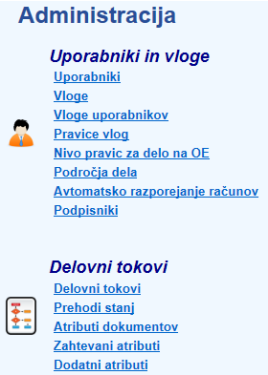
- »Stanja dokumentov«, kjer se bo v šifrantu lahko popravljalo nazive stanj, v kolikor bodo uporabniki to želeli;
- »Seznam nadomeščanj«, ki ga prikazuje Slika 25, kjer administrator lahko določi odsotnega uporabnika, namestnika in datum nadomeščanja (od–do). Zapise bo možno naknadno tudi urejati in brisati. Namestnik prevzame vse vloge in pravice odsotnega. V vseh dnevnikih bo za vsako spremembo evidentirano, da jo je vnesel namestnik.

Slika 25: Urejanje nadomeščanj

Vir: Podjetje za računalniški inženiring (2018).

Meni »Administracija«, prikazan na Sliki 26, bo namenjen uporabniku s pravico administratorja. Tukaj bo nastavljal pravice in vloge uporabnikov aplikacije.

Slika 26: Administracija uporabnika

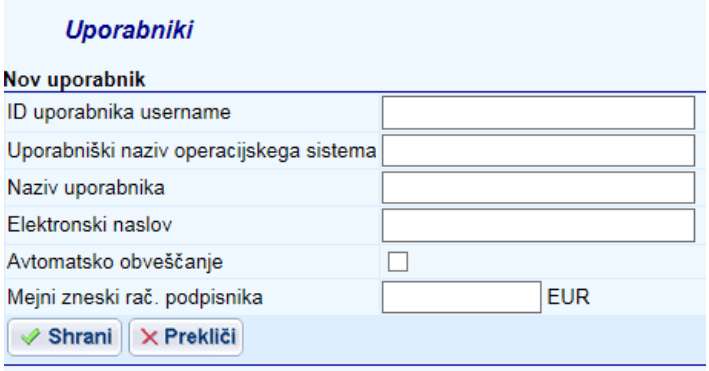


Vir: Podjetje za računalniški inženiring (2018).

Šifranti so naslednji:

- Uporabniki – v ta šifrant bo treba vnesti vsakega uporabnika, kar prikazuje Slika 27. Vnese se ime in priimek ter ime uporabnikovega operacijskega sistema (preverjanje pristnosti sistema Windows). Uporabniki bodo vrstično razvrščeni, lahko pa se jih bo sortiralo naraščajoče ali padajoče s pomočjo iskalnika.

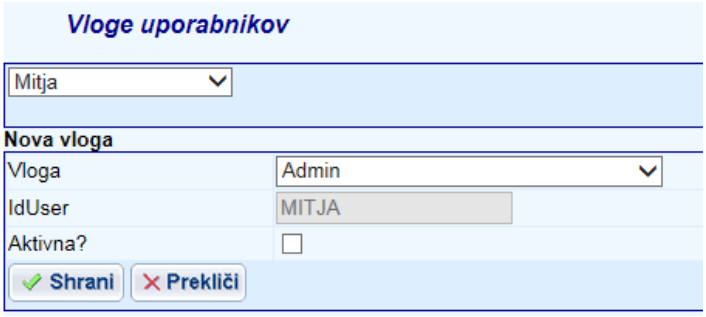
Slika 27: Vnos novega uporabnika



Vir: Podjetje za računalniški inženiring (2018).

- Vloge uporabnikov – tu se bo za prej vnesene uporabnike dodelilo vloge za delo v aplikaciji. Vnos nove vloge je prikazan na Sliki 28. Dodano bo tudi, ali lahko uporabnik z vlogo samo pregleduje ali lahko tudi ureja eNaročilnice. Vloge uporabnikov bodo vrstično razvrščene, lahko pa se jih bo sortiralo naraščajoče ali padajoče s pomočjo iskalnika.

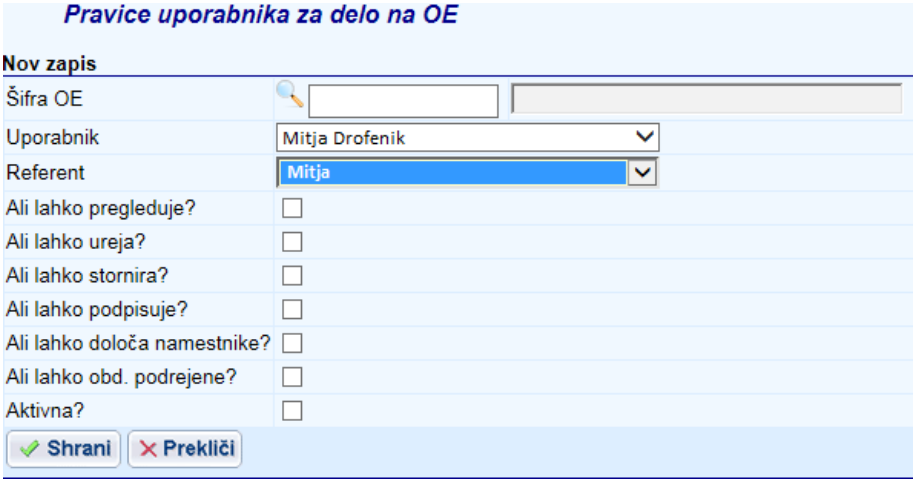
Slika 28: Vnos nove vloge



Vir: Podjetje za računalniški inženiring (2018).

- Slika 29 prikazuje šifrant, kjer se uporabnikom območne enote določi nivo pravic za delo v območni enoti. Tako bo lahko imel en uporabnik tudi več pravic za delo v različnih območnih enotah. Pravice uporabnikov bodo vrstično razvrščene, lahko pa se jih bo sortiralo naraščajoče ali padajoče s pomočjo iskalnika.

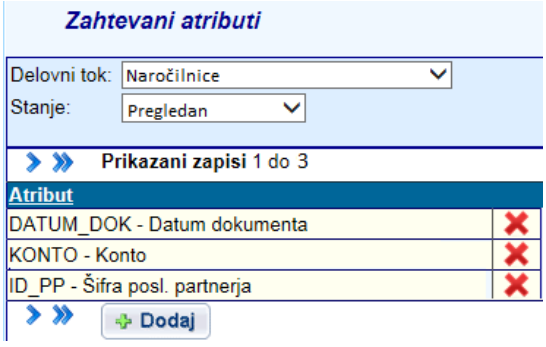
Slika 29: Dodajanje pravice uporabniku



Vir: Podjetje za računalniški inženiring (2018).

- Zahtevani atributi so tisti, ki jih je treba izpolniti pred pošiljanjem v naslednje stanje, in so prikazani na Sliki 30. Tukaj se bo poljubno dodajalo in odstranjevalo atribute, bodo pa ti predhodno že definirani glede na analizo. Atributi bodo vrstično razvrščeni, lahko pa se jih bo sortiralo naraščajoče ali padajoče s pomočjo iskalnika.

Slika 30: Zahtevani atributi



Vir: Podjetje za računalniški inženiring (2018).

Meni »Administrativno urejanje« bo namenjen administratorju, ki bo tu lahko urejal stanja dokumentov. Podatki bodo vrstično razvrščeni, lahko pa se jih bo sortiralo naraščajoče ali padajoče s pomočjo.

Na zavihku »Domov« je osrednji del namenjen informaciji o dodeljenih vlogah s številom eNaročilnic v vsaki vlogi. Zavihek »Domov« prikazuje Slika 31.

Slika 31: Zavihek Domov

Vir: Podjetje za računalniški inženiring (2018).

Ob kliku na vlogo se bo odprl meni z vsemi eNaročilnicami, ki čakajo na obdelavo v tej vlogi. Meni bo razdeljen na stolpce z vsemi pomembnimi podatki, ki jih bo mogoče urejati naraščajoče ali padajoče. Ob kliku na posamezni dokument se ta odpre na novi maski, kjer ga je mogoče urejati, če vloga to dopušča.

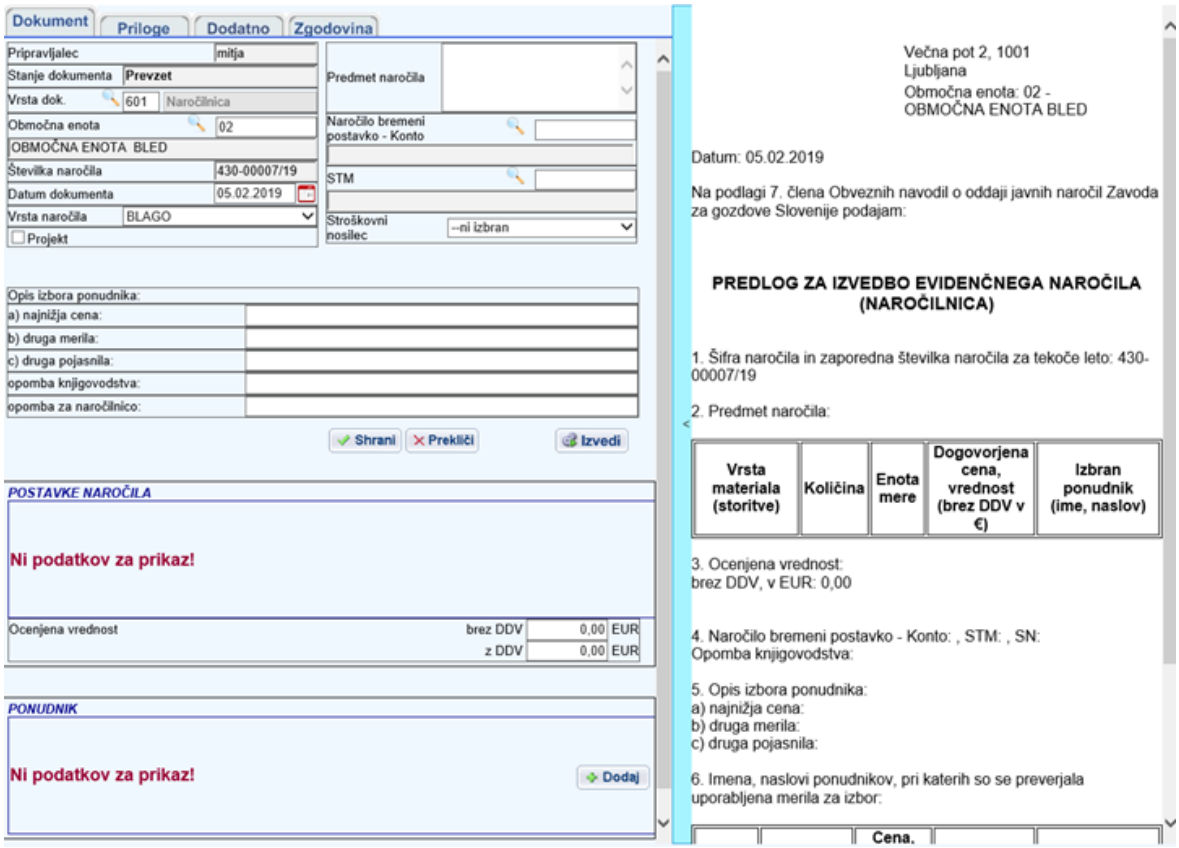
3.3.4 eNaročilnica v obdelavi

Maske za vnos eNaraočilnic so sestavljene iz 4 zavihkov:

- Dokument,
- Priloge,
- Dodatno,
- Zgodovina.

Zavihek »Dokument« je osnovna maska in bo razdeljen na dva dela, kot je tudi razvidno s Slike 32. Levi del bo namenjen vnosu podatkov eNaročilnice, na desnem pa se bo na podlagi vnesenih podatkov vizualizirala eNaročilnica. Desni del bo mogoče poljubno širiti in krčiti. Do maske bo uporabnik dostopal preko menija »Domov« ali preko menija »Ostali dokumenti«. Glede na vlogo bodo polja odklenjena ali zaklenjena. Vsak vnos bo treba shraniti z gumbom »Shrani«, obenem pa se bo vsaka sprememba tudi arhivirala. Za izvajanje akcij bo namenjen gumb »Izvedi«, ki bo odprl nabor mogočih akcij.

Slika 32: Zavihek Dokument



Vir: Podjetje za računalniški inženiring (2018).

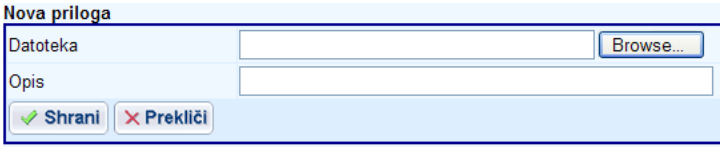
Zavihek »Priloge« bo namenjen dodajanju in urejanju prilog eNaročilnice. Zavihek »Priloge« prikazuje Slika 33. Uporabnik bo lahko vsako prilogo odprl s klikom na gumb »Slika« ter jo po potrebi natisnil ali shranil. Priloge so prikazane vrstično s podatkom o tem, za kakšen tip priloge gre. Prilogo se bo lahko izbrisalo s klikom na križec. Zaključenim dokumentom bo onemogočeno brisanje prilog.

Slika 33: Zavihek Priloge

Vir: Podjetje za računalniški inženiring (2018).

Z gumbom »Dodaj« bodo lahko dodane nove priloge, in sicer z vpisom poti do datoteke in opisom za novo prilogo. Vnos bo potrjen z gumbom »Shrani«. Dodajanje prikazuje Slika 34.

Slika 34: Nova priloga

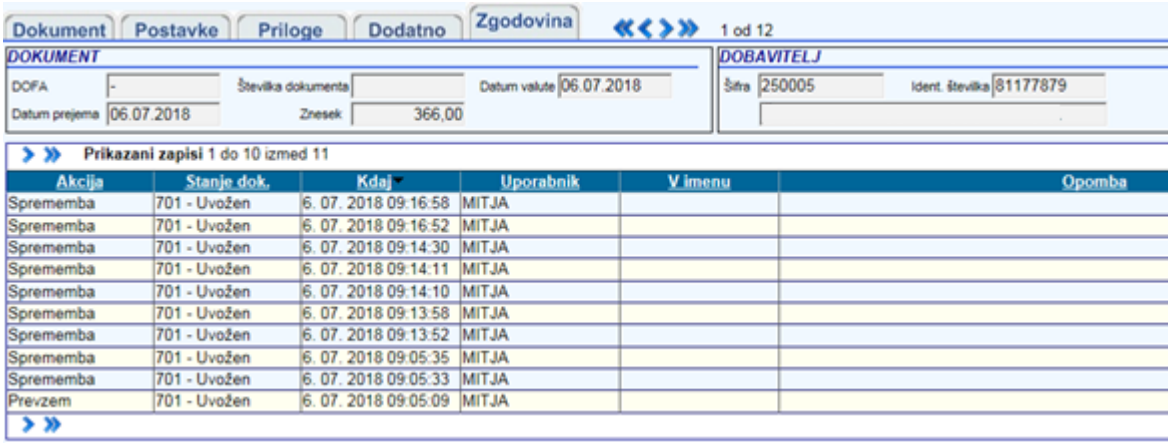


Vir: Podjetje za računalniški inženiring (2018).

Zavihek »Dodatno« trenutno nima funkcionalnosti, lahko pa se po potrebi dopolni glede na naročnikove želje.

Zavihek »Zgodovina«, kot ga prikazuje Slika 35, bo namenjen pregledovanju postopkov in sprememb, uveljavljenih na eNaročilnici. V zgornjem delu bodo prikazani osnovni podatki eNaročilnice, v srednjem delu pa seznam sprememb.

Slika 35: Zavihek Zgodovina



Akcija	Stanje dok.	Kdaj	Uporabnik	V imenu	Opomba
Sprememba	701 - Uvožen	6. 07. 2018 09:16:58	MITJA		
Sprememba	701 - Uvožen	6. 07. 2018 09:16:52	MITJA		
Sprememba	701 - Uvožen	6. 07. 2018 09:14:30	MITJA		
Sprememba	701 - Uvožen	6. 07. 2018 09:14:11	MITJA		
Sprememba	701 - Uvožen	6. 07. 2018 09:14:10	MITJA		
Sprememba	701 - Uvožen	6. 07. 2018 09:13:58	MITJA		
Sprememba	701 - Uvožen	6. 07. 2018 09:13:52	MITJA		
Sprememba	701 - Uvožen	6. 07. 2018 09:05:35	MITJA		
Sprememba	701 - Uvožen	6. 07. 2018 09:05:33	MITJA		
Prevzem	701 - Uvožen	6. 07. 2018 09:05:09	MITJA		

Vir: Podjetje za računalniški inženiring (2018).

3.4 Ekonomska upravičenost informacijske rešitve

Projekt pa poleg uspešne izvedbe zahteva tudi finančno podporo. V magistrskem delu ta vidik obravnavam kot zadnji, čeprav v praksi ni med zadnjimi, saj teče vzporedno z vsemi ostalimi aktivnostmi.

3.4.1 Analiza stroškov in koristi

Analiza stroškov in koristi oziroma CBA (ang. Cost Benefit Analysis) je postopek, s katerim se v prvi fazi ugotavlja, v drugi vrednoti ter v tretji fazi medsebojno primerja stroške in koristi podjetja z uvedbo informacijske rešitve. Osnovni cilj je ugotoviti, ali ima projekt večje koristi kot stroške. Z analizo stroškov in koristi želimo preveriti upravičenost vpeljave nove informacijske rešitve (Vojnovič, 2016).

Investicija se interpretira kot poraba finančnih sredstev v sedanjosti za pričakovano prihodnjo korist. Z ekonomskega vidika je torej načrtovan kapitalski izdatek v upanju na večjo količino denarnih prilivov v prihodnosti (Kavčič, 1996, str. 94–117).

Analiza stroškov in koristi je navadno sestavljena iz spodnjih faz (Tajnikar, 2003):

- finančna analiza z upoštevanimi finančnimi kazalniki,
- ekonomska analiza z upoštevanimi ekonomskimi kazalniki,
- analiza občutljivosti in tveganja.

Če želimo analizo uporabiti v praksi, je treba upoštevati probleme, ki lahko nastanejo, saj je jedro analize stroškov in koristi dokaj preprosto. Analizo se poveže s praktičnimi problemi, saj le tako lahko izračunamo današnje in prihodnje porabe za natančno oceno stroškov in koristi izvedbe projekta informacijske rešitve (Tajnikar, 2003).

Finančna analiza se pri CBA uporablja za izračun kazalnikov finančnih rezultatov investicije. Vrednotena je s strani investitorja, saj tega zanima, kako in kdaj se povrne vloženi kapital (Koda - svetovanje).

Ekonomska analiza je vrednotenje vložkov investicije. Za izhodišče se upošteva finančna analiza. Pri tem je navadno treba izvesti davčne popravke in popravke zaradi eksternalij (Vojnovič, 2016).

Tabela 2 prikazuje opis sedanjega stanja in stanja z informacijsko rešitvijo.

Tabela 2: Stanje brez in z informacijsko rešitvijo

Opis sedanjega stanja	Stanje z informacijsko rešitvijo
Čakanje na nadaljno izpolnjevanje in potrditev	Takojšna obvešečnost ter nadzor nadrejenih
Veliko birokracije	Manj pravil, ki so integrirana v aplikacijo
Fizično arhiviranje	Elektronski arhiv
Težja izvedba nadomeščanja	Z možnostjo nadomeščanja
Razlike med naročilnicami	Enotno izpolnjevanje

Vir: Lastno delo.

Ko bo informacijska rešitev v uporabi, torej nameščena na produkcijski strežnik, bodo koristi naslednje:

- hitro pošiljanje in obveščanje uporabnikov v procesu obdelave naročilnice,
- hitrejša obdelava v času odsotnosti uporabnikov,
- elektronski arhiv ter hitrejše iskanje že izvedenih naročil.

To vse pa na drugi strani tudi zmanjša:

- čas, ki so ga zaposleni potrebovali za obdelavo naročilnic,
- arhiv in pomeni manj dela z arhiviranjem,
- napake pri izpolnjevanju zaradi kontrol, vgrajenih v informacijski sistem.

Pričakujejo pa se tudi negativni vplivi (stroški) zaradi vpeljave eNaročilnic:

- večji stroški za pavšalno nadomestilo uporabe ePoslovanja,
- začetno uvajanje in nastavitve modula.

3.4.2 Donosnost naložbe

Investitor se odloči za naložbo, saj pričakuje, da se bo vloženi denar čim prej obrestoval. Donosnost naložb oziroma ROI (ang. Return on Investment) je merilo uspešnosti, ki se uporablja za ocenjevanje učinkovitosti naloženih sredstev. Donosnost naložbe meri dobljeni znesek donosa naložbe v primerjavi z vsemi stroški naložbe. Za izračun ROI se koristi naložbe delijo s stroški naložbe. Rezultat se izrazi z odstotkom oziroma razmerjem (Return on Investment).

Število denarnih enot, pridobljenih z naložbo, v odvisnosti od začetnega vložka predstavlja donos. Podatek kot tak nam ne pove veliko, v kolikor nimamo vseh podatkov o naložbi, zato je na podlagi podatka o donosu težko sklepati, ali je bila naložba dobra ali slaba. Pri naložbi 100 evrov je donos prav tako 100 evrov dober, pri naložbi 1000 evrov pa je enak donos 100 evrov slab. Pomemben pa je tudi čas; če naložba po 1 letu vrne 100 evrov, je donos dober, če pa za to potrebuje 10 let, je donos slab (Daves, Brigham & Brigham, 2018, str. 20).

V izogib zgornjim slabostim lahko pri računanju donosa naložbe upoštevamo formulo s Slike 36, ki donos opisuje kot odstotek povečanja oziroma zmanjšanja preučevane naložbe v preučevanem obdobju v primerjavi s preteklim obdobjem (Skok in drugi, 2004, str. 36).

Slika 36: Enačba za izračun ROI

$$Kapitalska\ donosnost = \frac{Cena_t - Cena_{t-1}}{Cena_{t-1}}$$

Vir: Berk Skok in drugi (2004).

Kjer je:

- Cena $_{t,}$ je cena pridobljena iz naložbe;
- Cena $_{t-1,}$ je cena vloženega kapitala.

Za izračun donosnosti je tako treba poznati vrednost začetnega kapitala ter dobiček ali izgubo, ki jo naložba prinese.

V našem primeru izračun donosnosti ni enostaven. Ker gre v Izbranem zavodu za informatizacijo poslovnega procesa kjer bo produkt namenjen interni rabi se pričakuje največji prihranek pri času, ki ga zaposleni namenijo obdelavi naročilnic Druge prihranke predstavljajo tudi:

- dokumentacija (papir),
- tiskalnik (tiskanje),
- fizično arhiviranje (čas, porabljen za zlaganje v fascikle),
- manjše zamude pri naročanju.

Ker imamo pet vlog, je za to potrebnih pet različnih oseb. Z uvedbo eNaročilnice je predviden čas izpolnjevanja ene eNaročilnice dve uri in trideset minut. Predviden čas delimo s številom oseb in pridemo do zaključka, da vsaka oseba porabi trideset minut. Ocenjujemo, da bodo eNaročilnice zmanjšale čas obdelave za polovico, zato lahko sklepamo, da je pred uredbo vsaka oseba za izpolnjevanje naročilnice porabila eno uro (Izbrani zavod, 2018).

Če za primer vzamemo veljavno bruto plačo sodelavca za finančne zadeve, ki spada v dvaintrideseti plačni razred (Katalog funkcij, delovnih mest in nazivov, 2018), ta znaša 1485,46 € (Veljavna plačna lestvica, 2016). Ob upoštevanju 40-urnega delovnika in dvaindvajsetih delovnih dni v mesecu dobimo bruto znesek ene ure, ki je enak 8,4375 €.

V enem mesecu vse območne enote skupaj v povprečju izdajo trideset naročilnic. Za vsako izdano naročilnico bo izbrani zavod prihranil 30 minut na osebo, kar znese za vse sodelujoče v delovnem toku dve uri in trideset minut. Preračunano v evre je to 21,094 €.

Dobljeni znesek pomnožimo s trideset in dobimo 632,82 € prihranjenega denarja na račun dela v enem mesecu (Izbrani zavod, 2018).

K temu bi lahko prišteli še znesek stroškov za papir in tiskalnik, a je znesek zanemarljiv, zato bomo ostali samo pri prihranku dela.

Ker je bila dopolnitev manjšega obsega, je bilo okvirno zaračunanih dvestopetdeset delovnih ur, kar pri okvirni ceni 60 € na uro znese 15.000 € investicije.

Tako imamo začetno investicijo 15.000 € ter mesečni prihranek 632,82 €. Če uporabimo zgornjo enačbo in vstavimo vrednosti za eno leto, dobimo spodnji izračun (1).

$$ROI = (7.593,84 - 15.000) / 15.000 = -0,4937 \qquad (1).$$

Rezultat pomnožimo še s količnikom sto in dobimo –49,37 %. Investicija ima v prvem letu uporabe modula eNaročilnice 49,37 odstotno izgubo.

Če vstavimo podatke za 2 leti. dobimo spodnji rezultat (2).

$$ROI = (15.187,68 – 15000) / 15.000 = 0,0125 \quad (2).$$

Rezultat pomnožimo še s količnikom sto in dobimo 1,25 %, kar pa že pomeni profit izbranega zavoda glede na začetno investicijo.

V prihodnosti se pričakujejo še dodatne nadgradnje informacijske rešitve, zaradi spreminjanja internih pravil Izbranega zavoda. Prav tako lahko pride do spremenjenega krogotoka, ki bo lahko dodatno razbremenil zaposlene.

SKLEP

Za uspešen razvoj informacijske rešitve je treba izbrati ustrezno metodologijo razvoja in tehnologijo, na kateri bo rešitev slonela, saj to omogoča skrajšanje razvojnega časa in enostavnejše vodenje projekta. S pravilno kombinacijo zgornjih dveh spremenljivk se povečajo tudi prednosti naročnika, ki se na koncu kažejo kot rezultati zadovoljstva uporabnikov z rešitvijo.

Metodologije se delijo na dve skupini, in sicer na agilne, ki jih označujemo tudi kot gibke, ter na tradicionalne, imenovane tudi težke metodologije.

Agilne in tradicionalne metodologije imajo mnogo prednosti, vendar se je treba pri odločanju osredotočiti tudi na njihove slabosti, saj je izbira pravilne metodologije pomembna odločitev pri načrtovanju razvoja informacijske rešitve.

Za nekatera področja je ena izmed metodologij bolj primerna za razvoj kot druga, obstajajo pa tudi področja, kjer ena ali druga metodologija sploh nista primerni za razvoj informacijske rešitve. Tako lahko metodologije za razvoj informacijske rešitve delimo na primerne in neprimerne. Tudi ob izbiri neprimerne metodologije lahko pridemo do končnega rezultata, je pa za to treba vložiti več truda in sredstev. V našem primeru je bila zaradi specifičnosti uporabljena metodologija poenotenega procesa, ki je objektno usmerjena, njen temelj pa so inkrementi in iterativni razvoj. Projekt je bil razdeljen na več manjših projekov, pri čemer je vsak predstavljal del aplikacije, na koncu pa so bili združeni. Razvoj aplikacije je temeljil na orodju Oracle CASE, znotraj iteracij pa je bil uporabljen klasičen slapovni model. Ker so bile naročnikove želje sprva slabo definirane, se je na podlagi že izdelane rešitve pripravila nova analiza z novo oceno potrebnega dela za spremembe oziroma dograditve. V tem primeru se je aplikacijo naročniku pošiljalo postopoma, tam pa so jo lahko v vmesnih fazah testirali in sporočali njeno ustreznost ter morebitne spremembe, ki so bile nato realizirane v naslednji iteracijah.

Tradicionalna metodologija ni predstavljala optimalne rešitve, saj bi celotna izgradnja trajala predolgo, prav tako pa so se v času razvoja zahteve naročnika spreminjale, kar

pomeni, da rešitev po tradicionalni metodologiji ne bi ustrezala naročniku, saj v sklopu tradicinalne metodologije spremembe med izgradnjo niso predvidene.

Spletna aplikacija eNaročilnice je narejena v okviru tradicionalnih spletnih aplikacij s tehnologijo Microsoft ASP.NET, ki deluje na Microsoftovem spletnem strežniku IIS. Ogrodje za izgradnjo je .NET Framework 4.5.

Hitre spremembe v poslovnem okolju imajo v zadnjih letih velik vpliv na podjetja. Z razvojem informacijske tehnologije se odpirajo nove priložnosti, hkrati pa vse to podjetja sili, da svoje poslovanje prilagajajo novim razmeram in začnejo uporabljati nova digitalna orodja, kot je ePoslovanje. Z avtomatizacijo poslovnih procesov se ti tudi prenovijo, saj jih poskušajo čim bolj poenostaviti, da bi delovni tok potekal kar se da nemoteno.

Cilj magistrskega dela je bila analiza informacijske rešitve za uvedbo eNaročilnic v izbrani zavod, ki bo podjetju omogočala vodenje naročanja preko enotnega programa in s čim manj papirologije. Tako sta v magistrskem delu predstavljeni analiza in rešitev nove aplikacije za evidetiranje eNaročilnic.

eNaročilnice se bodo uporabljale v vseh območnih enotah. Vsa naročila večjega zneska bodo šla še v potrditev v centralno enoto v Ljubljani, kjer bo naročilo morala odobriti vodja računovodstva. Aplikacija uporablja že znan uporabniški vmesnik, saj v izbranem zavodu že uporabljajo aplikacijo z enako tehnologijo za likvidaturo prejetih računov, ki je prijazna do uporabnikov ter pregledna in enostavna za uporabo. eNaročilnice se vnaša na podlagi predhodno določenega delovnega toka in vsaka vloga ima predpisana polja preko zahtevanih atributov, ki morajo biti izpolnjena, sicer je uporabnik ne more poslati v naslednje stanje.

Kot analitik v podjetju za računalniški inženiring sem pri uvedbi nove informacijske rešitve za izbrani zavod sodeloval s projektantom ter vodjo projekta. Pripravljal sem analizo, usmerjal programerje ter na koncu rešitev testiral in sodeloval pri uvajanju uporabnikov. Zaradi specifičnosti je bila uporabljena metodologija poenotenega procesa. Bistvo te metodologije je objektna usmerjenost, njen temelj pa so inkrementi in iterativni razvoj, zato je bil projekt razdeljen na več manjših oziroma posameznih projekov, pri čemer je vsak predstavljal del aplikacije, na koncu pa so se združili. Aplikacija je bila razvita na podlagi analize trenutnega delovnega toka in v okviru tega je bil predlagan ter s strani naročnika tudi potrjen nov poenostavljen delovni tok, ki občutno zmanjša delo zaposlenih izbranega zavoda ter omogoča lažjo evidenco ter manj papirologije. V prvih iteracijah je bila razvita rešitev, ki je podpirala osnovni delovni tok, vse posebnosti, ki so vezane na posamezno vlogo, ter na koncu tudi vizualizacija in končno generiranje naročilnice v pdf-formatu pa so bili doprogramirani kasneje. Model poenotenega razvoja je bil tukaj izbran zato, ker naročnik ni imel definiranih vseh posebnosti, ki so zanj pomembne.

Pri projektu eNaročilnic se je pokazalo, kako nujno potrebno je sodelovanje pri analizi produkta vseh, ki bodo aplikacijo uporabljali. Namreč velikokrat ima vodstvo napačno predstavo, kako delovni proces v praksi poteka in ravno to se je zgodilo tudi v izbranemu zavodu. Po analizirani in sprogramirani prvi verziji eNaročilnic, ki jo je vodstvo potrdilo brez vpletenosti ključnih bodočih uporabnikov aplikacije, se je izkazalo, da na tak način ne bo mogoče uporabljati programa.

Zato smo se še enkrat sestali sodelovci iz Računalniškega inženiringa ter Izbranega zavoda, kjer so bili prisotni tudi bodoči uporabniki, ki so natančno predstavili trenutni delovni tok, ter kje si želijo izboljšave. Vse njihove želje so bile natančno analizirane ter predstavljene v novi rešitvi, ki je opisana v poglavju 3, kjer so spremenjene predvsem vnosne maske ter delovni tok. Delovni tok tako vsebuje več vpletenih zaposlenih ter več aplikacijskih kontrol, da natančno sledi internim pravilom Izbranega zavoda.

Pri analizi delovnih tokov smo predlagali nekatere izboljšave in poenostavitve trenutnega delovnega toka, vendar zaradi obstoječih internih pravil niso bile vse sprejete.

LITERATURA IN VIRI

1. Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2002). Agile software development methods. *Vtt*. Pridobljeno 30. avgusta 2018 iz https://www.vtt.fi/inf/pdf/publications/2002/P478.pdf

2. Alexandrou, M. (2017, 20. november). Dynamic Systems Development Model (DSDM) Methodology. *Infolific*. Pridobljeno 1. septembra 2018 iz https://infolific.com/technology/methodologies/dynamic-systems-development-model/

3. Ambler, S. (2005). *Feature Driven Development (FDD) and Agile Modeling*. Agile Modeling. Pridobljeno 1. septembra 2018 iz http://www.agilemodeling.com/essays/fdd.htm

4. Awad, M. A. (2005). *Comparison between Agile and Tradicional Software Development Methodologies.* Perth: School of Computer Science and software Engineering, The University of Western Australia.

5. Chowdhury, E., Bhowmik, A., Hasan, H. & Rahim, S. (2018, maj). *Analysis of the Veracities of Industry Used Software Development Life Cycle Methodologies.* ResearchGate. Pridobljeno 10. oktobra 2018 iz https://www.researchgate.net/publication/325311524_Analysis_of_the_Veracities_of_Industry_Used_Software_Development_Life_Cycle_Methodologies

6. Bajec, M. (brez datuma). Razvoj informacijskih sistemov. *Ucilnica fri*. Pridobljeno 25. avgusta 2018 iz https://ucilnica.fri.uni-lj.si/enrol/index.php?id=99

7. Bajec, M. & Krisper, M. (2003, januar). Agilne metodologije razvoja informacijskih sistemov. *REsarchGate*. Pridobljeno 30. avgusta 2018 iz https://www.researchgate.net/publication/242778119_AGILNE_METODOLOGIJE_RAZVOJA_INFORMACIJSKIH_SISTEMOV

8. Beck, K. (1999). *Extreme Programming Explained: Embrace Change.* Boston: Addison-Wesley Professional.

9. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S.,Schwaber, K., Sutherland, J., Thomas, D. (2001). *Manifesto for Agile Software.* Agilemanifesto. *Development*. Pridobljeno 30. avgusta 2018 iz http://agilemanifesto.org/

10. Berk Skok, A., Lončarski, I., Zajc, P., Krajnović, E. K., Deželan, S., Groznik, P. & Valentinčič, A. (2004). *Poslovne finance.* Ljubljana: Ekonomska Fakulteta.

11. Berners-Lee, T. (1998, 7. maj). *The World Wide Web: A very short personal history*. Pridobljeno 9. avgusta 2018 iz https://www.w3.org/People/Berners-Lee/ShortHistory.html

12. Boehm, B. (2002). Get ready for agile methods, with care. *Computer, 35*(1), 64−69, Lancaster University: IEEE.

13. Bos, B. (2016, 17. december). *A brief history of CSS until 2016*. W3. Pridobljeno 9. avgusta 2018 iz https://www.w3.org/Style/CSS20/history.html

14. Brewer, C. (2018, 1. oktober). *Future of ecommerce*. Pridobljeno 8. junija 2018 iz http://www.future-of-ecommerce.com/eight-trends-to-watch-in-2018/

15. Cerovšek, M. (1 2012). Vzvodi informatike za rast in razvoj. Dlib. *Organizacija, letnik 45*, 27–35. Pridobljeno 13. avgusta 2018 iz https://www.dlib.si/stream/URN:NBN:SI:DOC-OADTMRIF/77039e63-2ef6-4516-97a2-e39a4c6473a2/PDF

16. Chung, W., Chen, H.-c. & Jr, J. F. (2005). A Visual Framework for Knowledge Discovery on the Web: An Empirical Study of Business Intelligence Exploration. *Journal of Management Information Systems, 21*(4), 57–84.

17. Clifton, M. & Dunlap, J. (2003, 29. september). What Is DSDM? *Code project.* Pridobljeno 1. septembra 2018 iz https://www.codeproject.com/Articles/5097/What-Is-DSDM

18. Cockburn, A. & Highsmith, J. (2001). *Agile Software Development: The People Factor*. *34*(11). Lancaster University: IEEE.

19. File-extensions. (2018, 12. februar). *css file extension*. Pridobljeno 15. avgusta 2018 iz https://www.file-extensions.org/css-file-extension

20. Čufer, B. (2008, april). Analiza izvajanja projektov prenove poslovnih procesov v Sloveniji : diplomsko delo visokošolskega strokovnega študija. Univerza v Mariboru, Fakulteta za organizacijske vede. Pridobljeno s https://dk.um.si/IzpisGradiva.php?lang=slv&id=29800

21. Daves, P., Brigham, E. & Brigham, D. (2018). *Intermediate Financial Management.* Boston: Cengage Learning Inc.

22. Diehl, D. (2013, 18. april). What is a Single Page Application? *Eguetech*. Pridobljeno 13. avgusta 2018 iz https://www.seguetech.com/what-is-a-single-page-application/

23. Fowler, M. (2005). The New Methodology. *Martinfowler*. Pridobljeno 2. septembra 2018 iz https://www.martinfowler.com/articles/newMethodology.html

24. 360logica Software Testing Services. (2015, 24. april). *General Trends in Agile Approaches*. Pridobljeno 10. oktobra 2018 iz https://www.360logica.com/blog/general-trends-in-agile-approaches/

25. Gradišar, M. (2017). *Computer Security Fundamentals*. Pridobljeno 2. avgusta 2018 iz http://studentnet.ef.uni-lj.si/CustomHandler/DownloadFile.ashx?fid=75933

26. Gradišar, M., Jaklič, J. & Turk, T. (2007). *Osnove poslovne informatike.* Ljubljana: Ekonomska fakulteta.

27. Gradišar, M., Jaklič, J., Damij, T. & Baloh, P. (2005). *Osnove poslovne informatike.* Ljubljana: Ekonomska fakulteta.

28. Green, B. & Shyam, S. (2013). *AngularJS.* Sebastopol: O'Reilly Media.

29. Hannah, J. (2015, 9. april). Choosing the Right JavaScript Framework for the Job. *Lullabot*. Pridobljeno 15. avgusta 2018 iz https://www.lullabot.com/articles/choosing-the-right-javascript-framework-for-the-job

30. Haviv, A. Q. (2014). *MEAN Web Development.* Birmingham: Packt Publishing Ltd.

31. Highsmith, J. (2002). *Agile Software Development Ecosystem.* Boston: Addison-Wesley.

32. Highsmith, J. (2011, 7. julij). Don’t Plan, Speculate. *Jimhighsmith*. Pridobljeno 9. oktobra 2018 iz http://jimhighsmith.com/dont-plan-speculate/

33. Hvala, D. (2011). *Okretno programiranje.* Monitor Pro. Pridobljeno 26. avgusta 2018 iz https://www.monitor.si/media/objave/dokumenti/2016/11/17/monitorpro1101.pdf

34. Itinfo. (2018). *Software Development Methodologies.* Pridobljeno 19. avgusta 2018 iz http://www.itinfo.am/eng/software-development-methodologies/

35. Izbrani zavod. (2018). *Poslovno poročilo izbranega zavoda (interno gradivo).*Ljubljana: Izbrani zavod

36. Jerman-Blažič, B., Klobučar, T., Perše, Z. & Nedeljković, D. (2001). *Elektronsko poslovanje na internetu.* Ljubljana: GV založba.

37. Kalakota, R. & Whinston, A. (1997). Electronic Commerce. *A Manager's Guide*. (16).

38. Kavčič, S. (1996). *Ekonomika kmetijskega gospodarstva.* Ljubljana: Biotehniška fakulteta.

39. Kettunen, P. (2009). Adopting Key Lessons from Agile Manufacturing to Agile Software Product Development – A Comparative Study. *Technovation, 29(6–7),* 408–422.

40. Kochhar, M. (brez datuma). *Small business*. Pridobljeno 8. avgusta 2018 iz https://smallbusiness.yahoo.com/advisor/5-experts-future-e-commerce-155024006.html

41. Koda - svetovanje. (brez datuma). *Cost-benefit analize*. Pridobljeno 7. oktobra 2018 iz http://www.koda-svetovanje.si/izdelava-dokumentacije/11-storitve/33-cost-benefit-analize

42. Kosi, T. (2010). *Poslovni procesi.* MIZS. Pridobljeno 15. avgusta 2018 iz http://www.mizs.gov.si/fileadmin/mizs.gov.si/pageuploads/podrocje/vs/Gradiva_ESS/Impletum/IMPLETUM_61EKONOMIST_Posl_procesi_Kosi.pdf

43. Kovačič, A. (1998). *Informatizacija poslovanja.* Ljubljana: Ekonomska Fakulteta.

44. Kovačič, A. & Vukšić, B. (2005). *Management poslovnih procesov: prenova in informatizacija poslovanja s praktičnimi primeri*. Ljubljana: GV Založba.

45. Krisper, M., Rupnik, R., Rožanec, A., Bajec, M., Osojnik, R. & Tomažič, R. (2003). *Enotna metodologija razvoja informacijskih sistemov*: Strateško planiranje. Ljubljana: Vlada Republike Slovenije, Center Vlade RS za informatiko.

46. Krum, R. (2016, 11. maj). The Growth of the Internet of Things. *Cool Infographics.* Pridobljeno 1. oktobra 2018 iz http://coolinfographics.com/blog/2016/5/11/the-growth-of-the-internet-of-things.html

47. Mikowski, M. & Powell, J. (2013). *Single Page Web Applications.* Greenwich: Manning Publications Co..

48. Ministrstvo za javno upravo. (9. junij 2018). *Katalog funkcij, delovnih mest in nazivov* Pridobljeno 9. oktobra 2018 iz http://www.mju.gov.si/fileadmin/mju.gov.si/pageuploads/JAVNA_UPRAVA/DPJS/Katalog/Katalog_FDMN_15.9.2018.pdf

49. Nahtigal, F. (2010). Pridobljeno 25. julija 2018 iz http://www.mizs.gov.si/fileadmin/mizs.gov.si/pageuploads/podrocje/vs/Gradiva_ESS/Impletum/IMPLETUM_284POSLOVNI_Elektronsko_Nahtigal.pdf

50. Naveen. (2015, 2. julij). What is Incremental Model in software testing and what are advantages and disadvantages of Incremental Model. *Testingfreek.* Pridobljeno 11. oktobra 2018 iz http://testingfreak.com/incremental-model-software-testing-advantages-disadvantages-incremental-model/

51. Ošlak, D. (2005, februar). Varnost elektronskega poslovanja v slovenskem bančništvu : magistrsko delo. Univerza v Ljubljani, Ekonomska fakulteta. Pridobljeno od http://www.cek.ef.uni-lj.si/magister/oslak430.pdf

52. Peyrott, S. (2017, 16. januar). *A Brief History of JavaScript*. Auth0. Pridobljeno 9. avgusta 2018 iz https://auth0.com/blog/a-brief-history-of-javascript/

53. Podjetje za računalniški inženiring. (2008). Prodajna dokumentacija. *Finančne aplikacije (interno gradivo).* Ljubljana: Podjetje za računalniški inženiring.

54. Podjetje za računalniški inženiring. (2010). *OBR_analiza* (interno gradivo). Ljubljana: Podjetje za računalniški inženiring.

55. Podjetje za računalniški inženiring. (2011). Opis produktne dokumentacije (interno gradivo). Ljubljana: Podjetje za računalniški inženiring.

56. Podjetje za računalniški inženiring. (2014). *Standardni predpis št. 01* (interno gradivo). Ljubljana: Podjetje za računalniški inženiring.

57. Podjetje za računalniški inženiring. (2017). *Analiza eNaročilnic*. Ljubljana: Podjetje za računalniški inženiring.

58. Podjetje za računalniški inženiring. (2018). *Analiza Dashbord*. Ljubljana: Podjetje za računalniški inženiring.

59. Powell-Morse, A. (2017, 20. januar). *Extreme Programming: What Is It And How Do You Use It?* Airbrake. Pridobljeno 10. oktobra 2018 iz https://airbrake.io/blog/sdlc/extreme-programming

60. Pucihar, A. (2018, 2. avgust). Pridobljeno 6. avgusta 2018 iz http://ecenter.fov.uni-mb.si/ecomSLO/Studenti/Predmeti/Prezentacije/e-Poslovanje1.ppt

61. Razgoršek, J. & Potočar, Z. (2009). Elektronsko poslovanje. Pridobljeno 25. julija 2018 iz http://www.mizs.gov.si/fileadmin/mizs.gov.si/pageuploads/podrocje/vs/Gradiva_ESS/Impletum/IMPLETUM_285POSLOVNI_Elektronsko_Potocar.pdf

62. Investopedia. (brez datuma). *Return on Investment*. Pridobljeno 7. oktobra 2018 iz https://www.investopedia.com/terms/r/returnoninvestment.asp

63. Rummler, G. A. & Brache, A. P. (1995). *Improving Performance: How to Manage the White Space in the Organization Chart.* San Francisco: Jossey-Bass.

64. Rupnik, R. (brez datuma). *Osnove informacijskih sistemov.* Pridobljeno 19. avgusta 2018 iz http://freeweb.siol.net/dragmann/OIS_predavanja_v2004_11.pdf

65. *SASS_REFERENCE*. (2015, 19. september). Sass-lang. Pridobljeno 15. avgusta 2018 iz http://sass-lang.com/documentation/file.SASS_REFERENCE.html

66. Sathram, B. (2014, 1. maj). *It's all about being Agile*. Pridobljeno 8. oktobra 2018 iz Agilegod: http://agilegod.blogspot.com/2014/05/crystal.html

67. Sava, G. (2017, 16. avgust). *Software development methodologies*. Supinfo. Pridobljeno 10. oktobra 2018 iz https://www.supinfo.com/articles/single/5027-software-development-methodologies

68. Schwaber, K. (1996). *Controlled Chaos: Living on the Edge.* Pridobljeno 1. septembra 2018 iz http://static1.1.sqspcdn.com/static/f/447037/6485970/1270926057073/Living+on+the+

69. Shore, J. & Warden, S. (2008). *The Art of Agile Development.* Sebastool: O'Reilly Media.

70. Sušnik, J. (2017). Razvoj večplatformnih aplikacij s pomočjo spletnih tehnologij za področje TV sporedov. *Repozitoriji*. Pridobljeno 9. avgusta 2018 iz https://repozitorij.uni-lj.si/Dokument.php?id=108347&lang=slv

71. Škrlec, I. (2002, 5. maj). *E-poslovanje in uporaba formata XML*. Ljubljana: Finance.

72. Tajnikar, M. (2003). *Mikroekonomija s poglavji iz teorije cen*. Ljubljana: Ekonomska Fakulteta.

73. Thurrott, P. (2017, 14. maj). *This is What Microsoft Said About Progressive Web Apps at Build*. Thurrott. Pridobljeno 9. avgusta 2018 iz https://www.thurrott.com/windows/windows-10/116101/microsoft-said-progressive-web-apps-build

74. Turban, E. (2009, oktober). *Electronic Commerce: A Managerial Perspective* 2010. (6), str. 7−8. New York: Prentice Hall.

75. Turban, E., McLean, E. & Wetherbe, J. (1999). *Information Technology for Management – Making Connections for Strategic Advantage*. New York: John Wiley & Sons.

76. Valdarrama, S. L. (2014, 15. oktober). *Is a Single-Page Application what you really need?* Shiftedup. Pridobljeno 15. avgusta 2018 iz https://www.shiftedup.com/2014/10/15/is-a-single-page-application-what-you-really-need

77. *Veljavna plačna lestvica*. (2016, 1. september). Ministrstvo za javno upravo Pridobljeno 9. oktobra 2018 iz http://www.mju.gov.si/si/delovna_podrocja/place_v_javnem_sektorju/veljavna_placna_lestvica/

78. Villagomez, E. (2009, 16. oktober). *An overview of the rational unified process (RUP).* Slideshare. Pridobljeno 9. oktobra 2018 iz https://www.slideshare.net/ERICEV/rup-2248862

79. Vojnovič, D. (2016, oktober). *Analiza stroškov in koristi*. Ljubljana: Ministrstvo za okolje in prostor.

80. Warcholinski, M. (2018, 6. april). *Differences Between Lean, Agile and Scrum*. Brainhub. Pridobljeno 9. oktobra 2018 iz https://brainhub.eu/blog/differences-lean-agile-scrum/

81. Williams, L. A. & Cockburn, A. (2003). Agile software development: it's about feedback and change. *Computer, 36*, 39-43.

82. Wong, S. & Anand, V. (2013, 23. januar). *Hardwarezone*. Pridobljeno 7. avgusta 2018 iz https://www.hardwarezone.com.sg/feature-introduction-near-field-communication-nfc-technology/nfc-usage-beyond-e-commerce

83. Yu, B. L., Wooi, K. L., Wai, Y. T. & Soo, F. T. (2012). *Software Development Life Cycle AGILE vs Traditional Approaches.* Semanticscholar. Pridobljeno 19. avgusta

2018 iz https://pdfs.semanticscholar.org/69b1/9ddc8a578f4c63d1dfe15252a465ee12fe5d.pdf

84. Zornada, L. (2002). Razvoj informacijskega sistema – od strateškega načrta do realizacije. *Repozitorij*. Pridobljeno 19. avgusta 2018 iz https://repozitorij.upr.si/IzpisGradiva.php?lang=slv&id=1974