UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

MAGISTRSKO DELO

RAZVOJ INFORMACIJSKEGA SISTEMA ZA UPRAVLJANJE Z LIKVIDNOSTNIM TVEGANJEM V FINANČNI INSTITUCIJI

Ljubljana, maj 2016 PETER STARBEK

IZJAVA O AVTORSTVU

Podpisani(-a) PETER STARBEK, študent/-ka Ekonomske fakultete Univerze v Ljubljani, avtor/-ica predloženega dela z naslovom RAZVOJ INFORMACIJSKEGA SISTEMA ZA UPRAVLJANJE Z LIKVIDNOSTNIM TVEGANJEM V FINANČNI INSTITUCIJI, pripravljenega v sodelovanju s svetovalcem DR. MIRO GRADIŠARJEM.

IZJAVLJAM

1. da sem predloženo delo pripravil/-a 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/-a, 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/-a 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/-a 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

UVOD

Učinkovito in stabilno poslovanje, ki temelji na uravnovešanju denarnega toka, je stalni izziv vsakega podjetja. Vsako podjetje je lahko v težavah, če nima zadostnega denarnega toka. Ta ima še posebej veliko veljavo v finančnih institucijah, saj se prek njih pretaka denar kot gonilna sila gospodarstva (Brigham & Ehrhardt, 2004, str. 13). Brez njih na finančnih trgih ne bi prihajalo do učinkovite alokacije finančnih sredstev od tistih, ki varčujejo, do tistih, ki imajo poslovne priložnosti (Prohaska, 2004, str. 10).

Za dolgoročno uspešno poslovanje finančnih institucij je ključno, da so sposobne dosledno obvladovati tveganja oziroma da so uspešne pri upravljanju s tveganji, s katerimi se srečujejo pri svojem finančnem posredništvu (Allen, 2003, str. 21). Upravljanje denarnega toka oziroma upravljanje likvidnostnega tveganja pomeni upravljanje enega izmed ključnih tveganj in je bistvo zdravega poslovanja vsakega podjetja. Z upravljanjem točno vemo, koliko razpoložljivih tekočih sredstev potrebujemo, koliko jih potrebujemo za razširitev poslovanja, koliko in po čem jih lahko pridobimo na trgu. To je pomemben dejavnik pri zagotavljanju konkurenčnosti in uspešnosti podjetja. Likvidnost oziroma likvidno poslovanje pomeni sposobnost podjetja poravnavati vse obveznosti ob zapadlosti. Če tega ni zmožno, potem rečemo, da je podjetje nelikvidno in mora iskati zunanje vire financiranja, če sploh želi ostati v poslu.

Pomemben dejavnik za podjetja pri doseganju konkurenčne prednosti je informacijska tehnologija. S pomočjo informacijske tehnologije, reorganizacije dela (prenove poslovanja) in odprave nepotrebne birokracije lahko dosežemo krajše izvajalne čase delovnih procesov (Gradišar & Resinovič, 2001). Pravilno odločanje je mogoče samo, če so na voljo točne, pravočasne in transparentne informacije. Podjetja, ki hočejo preživeti v poslovnem svetu, si morajo zato zagotoviti takšen informacijski sistem, ki jim bo to omogočal. Če tak informacijski sistem ne obstaja, ga je treba zasnovati in razviti.

Razvoj informacijskega sistema je dolgoročen investicijski poseg, pri katerem se mora upoštevati značilnosti okolja, uporabniške potrebe in zahteve, analizo stroškov, investicijsko politiko itd. Pri razvoju informacijskih sistemov uporabljamo različne metodologije. Metodologija je zbirka postopkov, tehnik, orodij in dokumentacijskih pripomočkov, ki razvijalcem pomaga pri njihovih naporih za implementacijo novega informacijskega sistema. Metodologija je sestavljena iz faz, ki se lahko še naprej delijo na podfaze. Te vodijo razvijalce sistema v njihovi izbiri tehnik, ki bi lahko bile primerne v vsaki fazi projekta, in jim pomagajo načrtovati, upravljati, nadzirati in izvajati projekte izgradnje informacijskih sistemov (Avison & Fitzgerald, 2003, str. 528).

Zaradi velikega nabora metodologij razvoja programskih rešitev in projektnega vodenja, njihovih prednosti ter slabosti v posameznih okoliščinah se podjetja težko odločajo, katero izbrati. V precejšnji meri je še vedno prisotno razvijanje programskih rešitev po

zaporednem modelu (angl. *Waterfall model*), vendar pa se ga zaradi njegovih pomanjkljivosti vedno manj uporablja (Avison & Fitzgerald, 2003, str 39). Sodobne metodologije uvajajo agilne pristope, ki prinašajo dodano vrednost zaradi prožnosti, prilagodljivosti, konstruktivnega sodelovanja in popolne usmerjenosti v poslovno vrednost (Agile Manifesto, 2001).

Tudi za finančne institucije, kot so lizing hiše, je konkurenčna prednost bistvenega pomena. Le tiste lizing hiše, ki bodo imele dovolj razvit informacijski sistem, ki bo zagotavljal pravočasne ter kakovostne informacije, bodo imele možnost za uspeh in rast.

Pričujoče magistrsko delo je osredotočeno na reševanje te problematike. Izhaja iz trenutnega stanja informacijskega sistema izbranega lizing podjetja. Ta, tako kot velika večina lizing podjetij v Sloveniji, za svoje poslovanje uporablja informacijsko rešitev, ki jo je razvilo podjetje Globus Marine International, d. o. o. Rešitev omogoča upravljanje in spremljanje denarnega toka osnovne dejavnosti lizing podjetja – to je ponudbe lizinga strankam. To pa s stališča spremljanja in načrtovanja likvidnosti podjetja ni dovolj. Potrebno je celovito spremljanje vseh denarnih tokov, ki nastopajo med poslovanjem lizing podjetja. Tako poleg omenjenih tokov nastopajo tudi številni zunanji viri financiranja kot tudi drugi denarni tokovi, ki omogočajo nemoteno poslovanje.

Lizing podjetje sestavlja več odvisnih družb, zato je spremljanje likvidnosti podjetja še kompleksnejše. To je lahko zanimivo tako s stališča likvidnosti vsake odvisne družbe posebej kot tudi s stališča konsolidirane ravni pogleda likvidnosti vseh odvisnih družb skupaj.

Na trgu ne obstaja že izdelana rešitev, ki bi zadovoljivo reševala navedeno problematiko, zato se je v lizing podjetju sprejela odločitev za razvoj novega sistema.

Namen magistrskega dela je obravnava zasnove in razvoja informacijskega sistema za upravljanje z likvidnostnim tveganjem v izbranem lizing podjetju. Problematika in z njo povezana razvita rešitev pa se ne pojavljata samo znotraj izbranega lizing podjetja. Rešitev lahko posplošimo tudi na druge finančne institucije, kjer se prav tako srečujejo s podobnimi težavami upravljanja likvidnostnega tveganja. Rešitev tako lahko predstavlja predlog reševanja problematike s področja likvidnosti tudi vsem ostalim finančnim institucijam.

V magistrskem delu se posvetim zasnovi in razvoju programske rešitve za celovito upravljanje likvidnostih tveganj izbranega lizing podjetja, ki ga sestavlja več odvisnih družb. Pri tem utemeljim povode za razvoj rešitve, izbor ustrezne metodologije izmed množice metodologij razvoja ter predstavim razvoj rešitve skozi vse ključne faze izbrane metodologije. Delo zajema tudi opis in utemeljitev prilagoditev izbrane metodologije razvoja glede na posebnosti, ki so nastopale v opisanem projektu.

Projekt je preobsežen za enega človeka, zato je na njem delal tim ljudi, katerega član sem bil tudi sam. V največji meri sem sodeloval pri izbiri ustrezne metodologije dela, arhitekturni zasnovi rešitve kot tudi razvoju posameznih komponent rešitve, zato se v delu omejim predvsem na analizo tega področja rešitve.

V magistrskem delu sem si zadal **več ciljev**. Glavni cilj je analizirati informacijske potrebe in izdelati delujočo rešitev za spremljanje likvidnostnega tveganja lizing podjetja ter jo uvesti v podjetje. Poleg glavnega sem si zadal še naslednje cilje:

- analizirati obstoječe stanje procesa spremljanja likvidnosti v lizing podjetju,
- predstaviti osnove razvoja informacijskih sistemov,
- prilagoditi metodologije razvoja glede na zahteve in posebnosti projekta,
- podrobno predstaviti ključne faze razvoja v okviru izbrane metodologije,
- predlagati morebitne spremembe delovnih procesov,
- razviti prilagodljivo rešitev z možnostjo vključevanja novih modulov in poročil,
- predstaviti rešitev kot predlogo za reševanje podobne problematike v finančnih institucijah.

V začetnih poglavjih, ki vsebujejo predvsem teoretična izhodišča, uporabim pretežno metodo študija literature domačih ter tujih avtorjev, notranje vire družbe Avtenta.si, d. o. o., ter znanje, ki sem ga pridobil na podiplomskem študiju.

V drugem, praktičnem delu magistrskega dela pa uporabim metode kritične analize stanja pred uvedbo novega sistema ter sistemske analize, ki je osnovana predvsem na teoretičnih in praktičnih spoznanjih, ki sem jih pridobil v času razvoja.

Magistrsko delo je razdeljeno na sedem poglavij. Po uvodu v prvem poglavju predstavim vsebinsko področje, ki naj bi ga rešitev pokrivala. V drugem poglavju opredelim osnovne pojme in sestavine informacijskega sistema. Sledi tretje poglavje, kjer predstavim različne metodologije načrtovanja in razvoja informacijskih sistemov. V četrtem poglavju predstavim lizing podjetje, opišem analizo trenutnega stanja informacijskega sistema podjetja in podam želeno ciljno stanje informacijskega sistema podjetja. V petem poglavju opišem razvoj glavnine rešitve skupaj s periodičnimi dopolnitvami (nadgradnjami) prek vseh ključnih faz izbrane metodologije. Sledi šesto poglavje, analiziram ekonomiko rešitve z vidika stroškov, koristi in donosnosti investicije. V sedmem poglavju podrobno podam kritično analizo izbora metodologije, magistrsko delo pa zaključim s sklepnimi ugotovitvami.

1 PREDSTAVITEV PODROČJA UPRAVLJANJA Z LIKVIDNOSTNIM TVEGANJEM

1.1 Tveganja finančnih institucij

Tveganje lahko opredelimo kot vsako neželeno zmanjšanje gospodarskih koristi podjetja. Gre torej za možnost neprijetnega odmika od načrtovanih ciljev podjetja, ki je posledica negotovosti ocen razvoja dogodkov v prihodnosti kot tudi nepopolnosti informacij, ki jih vodstvo uporablja za sprejemanje poslovnih odločitev (Berk et al., 2005, str. 55).

Finančne institucije delujejo na finančnem trgu kot finančni posredniki. Od komitentov, ki imajo presežek sredstev, sprejemajo naložbena sredstva in jih posredujejo komitentom, ki potrebujejo finančna sredstva. Tako poslovanje finančne institucije je vedno povezano z različnimi vrstami tveganj. Pomembno je, da znajo ta tveganja identificirati in se proti njim ustrezno zaščititi. Proces upravljanja s tveganji je za finančne institucije ključnega pomena, saj že iz njihove primarne dejavnosti izhaja, da je sprejem določene stopnje tveganja neizogiben. Napačne odločitve glede izpostavljenosti različnim tveganjem lahko povzročijo izgubo ugleda, strank oziroma lahko celo ogrozi obstoj finančne institucije.

Finančne institucije se pri svojem delovanju srečujejo s tveganji, ki so značilna za bančno poslovanje, saj opravljajo dejavnost financiranja. Različni avtorji različno razvrščajo bančna tveganja. Saunders (2000, str. 103) v grobem loči med devetimi vrstami tveganja:

- tveganje spremembe obrestne mere,
- tržno tveganje,
- kreditno tveganje,
- zunajbilančno tveganje,
- tehnološko in operativno tveganje,
- tečajno tveganje,
- deželno tveganje,
- likvidnostno tveganje in
- tveganje plačilne sposobnosti.

Za namen magistrskega dela izpostavljam likvidnostno tveganje, saj se delo osredotoča le na opis razvoja rešitve upravljanja s to vrsto tveganja. Finančne institucije pa se pri opravljanju svoje dejavnosti srečujejo tudi z ostalimi vrstami tveganja, ki niso del vsebinskega poudarka magistrskega dela.

1.2 Pojem likvidnosti

Likvidnost finančne institucije pomeni sposobnost zagotovitve likvidnih sredstev za sprotno financiranje zapadlih obveznosti. Njeno obvladovanje je nujen in osnoven pogoj za delovanje in razvoj finančne institucije, saj ji omogoča njeno temeljno aktivnost posredovanja likvidnih sredstev. Likvidnost se odraža na vseh področjih poslovanja, saj ima vsak posel oziroma transakcija finančne institucije nanjo specifičen vpliv. Likvidnost finančne institucije je ustrezna, če pričakovani denarni pritoki zadoščajo za pokritje denarnih odtokov.

Bessis (2002, str. 7) definira tri stanja likvidnosti:

- pokrito likvidnost – sposobnost finančne institucije, da si lahko na trgu zagotovi potrebna likvidna sredstva za financiranje odlivov po tržnih cenah,
- opozorilno likvidnost – tveganje, da kratkoročna sredstva ne bodo zadoščala za pokritje kratkoročnih zahtevanih odlivov,
- ekstremno nelikvidnost – dolgoročna nesposobnost finančne institucije za poravnavo dospelih obveznosti, kar privede do nesolventnosti in stečaja finančne institucije.

Finančna institucija lahko zagotavlja likvidnost bodisi z zmanjševanjem ali s prestrukturiranjem investicij bodisi z zadolževanjem oziroma s pridobivanjem dodatnih virov. Premajhne likvidnostne rezerve pomenijo nevarnost, da ne bo uspela pravočasno poravnati obveznosti, medtem ko vzdrževanje prevelikih likvidnostnih rezerv pomeni breme z vidika donosnosti. Dve izmed najpomembnejših nalog znotraj finančne institucije sta zato analiziranje in postavitev ustreznih modelov optimiranja likvidnosti (Saunders, 2000, str. 358).

1.3 Likvidnostno tveganje

Likvidnostno tveganje je opredeljeno kot tveganje, ki se v finančnih institucijah pojavi zaradi izrednega povečanja povpraševanja po dvigu najbolj likvidnih oblik virov sredstev s strani upnikov (Saunders, 2000, str. 114). Do neusklajenih denarnih tokov prihaja zaradi transformacije kratkoročnih virov v dolgoročne naložbe, kar je primarna funkcija finančnih posrednikov. Ti so likvidni, kadar so sposobni vzdržati zmanjšanje obveznosti, poravnati zapadle obveznosti in povečevati finančna sredstva.

Glede na naravo in strukturo lahko opredelimo tri komponente likvidnostnega tveganja (Matz, 2007, str. 5):

- tveganje likvidnostne vrzeli, ki predstavlja pričakovano likvidnostno tveganje na podlagi razlike med denarnimi pritoki in odtoki,

- potencialno likvidnostno tveganje, ki pomeni tveganje, da bodo prihodnji dogodki zahtevali več likvidnostnih kapacitet kot običajno,
- tržno likvidnostno tveganje, ki je tveganje, da finančna institucija ne bo zmogla prodati želene količine likvidnih naložb po tržni ceni zaradi neprimerne tržne globine ali motenj na trgu.

Temelja značilnosti likvidnostnega tveganja sta asimetričnost in večdimenzionalnost (Metz, 2007, str. 5). Asimetričnost pomeni, da so potencialne negativne posledice izpostavljenosti likvidnostnemu tveganju precej večje od potencialnih učinkov. Večdimenzionalnost pomeni, da je likvidnostno tveganje treba obravnavati z vidika številnih okoliščin, ki vplivajo na potrebe po likvidnih sredstvih in na možnost njihove zagotovitve. To zahteva tudi primerna orodja za merjenje in upravljanje likvidnostnega tveganja.

Uravnavanje likvidnostnega tveganja je povezano s številnimi dejavniki. Te lahko v grobem razdelimo na:

- **notranje dejavnike**, ki izhajajo iz poslovanja finančne institucije, kar se odraža v strukturi naložb in obveznosti, ročnosti neusklajenosti denarnih tokov ter preveliki koncentraciji in odvisnosti od posameznih upnikov. Zato lahko razdelimo vpliv notranjih dejavnikov, ki se pojavijo na posamezni strani **bilance stanja finančne institucije** (Saunder, 2000, str. 358);
- **zunanje dejavnike**, ki jih predstavlja okolje, v katerem finančna institucija deluje. Pri tem so še posebej pomembni stabilnost gospodarskega okolja, bančne regulative in razvitost finančnih trgov.

1.4 Upravljanje likvidnostnega tveganja

Finančna institucija mora glede na stopnjo prevzetega tveganja ugotavljati, meriti, obvladovati in spremljati likvidnostno tveganje. Za njegovo učinkovito upravljanje mora imeti opredeljeno strategijo, saj lahko samo tako zagotavlja trajno in pravočasno poravnanje zapadlih obveznosti. Pri upravljanju tveganj poslovanja morajo finančne institucije upoštevati tudi predpise, standarde ter zakone države, v kateri se institucija nahaja. Tako že Zakon o bančništvu določa, da mora banka z likvidnostnim tveganjem oblikovati in izvajati politiko rednega upravljanja z likvidnostjo. Ta politika obsega (Uradni list RS, št. 131/06):

- načrtovanje pričakovanih znanih in morebitnih denarnih odtokov ter zadostnih denarnih pritokov zanje ob upoštevanju normalnega toka poslovanja in morebitnih položajev likvidnostnih kriz,
- redno spremljanje in upravljanje likvidnosti,

- opredelitev ustreznih ukrepov za preprečitev oziroma odpravo vzrokov za nastop nelikvidnosti in opredelitev drugih možnosti za take ukrepe.

Upravljanje z likvidnostnim tveganjem lahko razdelimo na dve ravni (Saunders, 2000, str. 362):

- prva raven zajema trenutno, dnevno naravnano operativno likvidnost. Običajno za to skrbita zakladništvo in oddelek obvladovanja tveganj,
- druga raven zajema dolgoročno strateško oziroma strukturno likvidnost, ki se ukvarja predvsem z možnostmi financiranja za zagotavljanje operativne likvidnosti. Zanjo običajno skrbijo vodstvo in posebne strokovne komisije.

Sinkey (1998, str. 248) podaja funkcije, ki jih skrbno upravljanje z likvidnostjo omogoča finančni instituciji:

- izpolnitev pogodbenih obveznosti,
- izogibanje izgubam zaradi prodaje naložb v sili in prodaje naložb »za vsako ceno«,
- lažji dostop do refinanciranja pri trgovanju s centralno banko,
- doseganje nižje višine premije za kreditno tveganje pri pridobivanju novih virov,
- vzdrževanje zaupanja komitentov, kar preprečuje nenadne odlive.

Saunders (2000, str. 367) cilje upravljanja z likvidnostjo definira kot:

- dnevno izpolnjevanje vseh obveznosti, povezanih z denarnimi odtoki,
- pridobivanje virov po tržnih cenah,
- izogibanje prisilnim preoblikovanjem naložb v likvidna sredstva,
- izpolnjevanje predpisov, ki urejajo področje likvidnosti in obvezne rezerve.

Prvi korak k obvladovanju likvidnostnega tveganja predstavlja njegovo merjenje, ki pa je težavno, saj je likvidnost odvisna od številnih nepredvidljivih in specifičnih dejavnikov. Tako zahteva visoko stopnjo predvidevanja prihodnjih denarnih tokov in poznavanja uravnavanja likvidnostne pozicije. Pri tem se uporablja številne metode in tehnike, ki se razlikujejo predvsem po kompleksnosti uporabe in obsegu potrebnih podatkov. Enostavni izračuni kazalnikov zahtevajo manj podrobnosti in dajejo preprosto oceno likvidnosti. Na drugi strani obstajajo zahtevnejši modeli, ki izračunavajo likvidnost na podlagi projekcije zapadlosti denarnih tokov ter ob upoštevanju verjetnosti, da se posamezni denarni tok zgodi.

Uspešno upravljanje likvidnostnega tveganja temelji na informacijah, ki jih zagotovijo posamezne tehnike merjenja likvidnostnega tveganja. Zaradi tega je treba vzpostaviti ustrezen informacijski sistem, ki zahteva sistematično zbiranje in hranjenje podatkov.

2 INFORMACIJSKI SISTEM

Visokokakovosten, pravočasen in dobro upravljan informacijski sistem je dandanes srce vsakega uspešnega podjetja. Za uspešno poslovanje podjetja je še kako pomembno, da pravi ljudje pridobijo pravo informacijo ob pravem času. To pomeni, da lahko pravilno vidijo stanje na različnih poslovnih področjih podjetja in na podlagi tega sprejemajo prave odločitve. Ena izmed pomembnejših zmogljivosti informacijskega sistema je organizacija informacij, potrebnih za učinkovito izvajanje operacij, uspešno upravljanje in zagotavljanje konkurenčne prednosti podjetja ter posledično uresničevanje njegovega poslanstva in poslovnih ciljev.

2.1 Opredelitev informacijskega sistema

Osnovna opredelitev informacijskega sistema pravi, da je informacijski sistem sistem, v katerem se generirajo, arhivirajo in pretakajo informacije (Gradišar & Resinović, 2001, str. 338).

Avison (v Avison & Fitzgerald, 2003, str. 19) meni, da informacijski sistem zbira, hrani, obdeluje in posreduje informacije, potrebne za delovanje podjetja. Zgrajen mora biti tako, da so informacije v njem uporabne in dostopne vsem, ki jih želijo uporabljati.

Turk (v Turk et al., 1987, str. 77) ga opredeljuje kot organizacijsko celoto medsebojno povezanih sestavin, ki imajo namen oblikovati informacije in hraniti podatke na različnih stopnjah njihovega obravnavanja.

Tudi Lesjak (2003, str. 19) informacijski sistem opredeli podobno, saj pravi, da primerno povezane (organizirane) sestavine omogočajo odvijanje informacijskega procesa, v katerem nastajajo informacije.

Če povzamem opredelitve informacijskega sistema, lahko rečem, da informacijski sistem v organizaciji predstavlja proces, ki zagotavlja informacije za bolj učinkovito upravljanje organizacije.

Srića (v Srića, Treven, & Pavlić, 1995, str. 20) pravi, da informacijski sistem najhitreje opišemo tako, da odgovorimo na naslednja tri vprašanja:

- kaj je cilj informacijskega sistema,
- katere so njegove funkcije in
- katere so njegove sestavine.

2.1.1 Cilj informacijskega sistema

Cilj vsakega informacijskega sistema je posredovati pravo informacijo na pravo mesto v organizaciji v pravem času in z minimalnimi stroški (Srića et al., 1995, str. 20).

Podobno ugotavlja Ključevšek (2001, str. 5), ki pravi, da je osnovni namen informacijskega sistema organizacije, da oskrbuje uporabnike z informacijami o preteklem in trenutnem delovanju ter o predvidenem obnašanju organizacije in njenega okolja oziroma da z minimalnimi stroški posreduje pravo informacijo v pravem času na pravo mesto v organizaciji.

V praksi takšnega cilja ni lahko uresničiti. Že na začetku je treba odgovoriti, katera informacija je prava. Tudi najbolj izkušen kader v podjetju pogosto ne zna povedati, katere podatke potrebuje pri reševanju določenega problema. Pravilno opredeliti problem in iz tega izpeljati informacijske potrebe ni lahka naloga. Vprašanje rešujemo predvsem z višanjem splošne ravni informacijske pismenosti.

2.1.2 Funkcije informacijskega sistema

Informacijski sistem opravlja štiri temeljne funkcije (Srića et al., 1995, str. 20):

- zbiranje podatkov,
- obdelavo podatkov,
- hranjenje podatkov ter informacij in
- posredovanje podatkov ter informacij uporabnikom.

Pri zbiranju podatkov se ukvarjamo predvsem z vprašanji, kot so, katere izvore uporablja sistem, od kod prihajajo njegove vhodne sestavine in kako bomo pripravili, zbirali ter vnesli potrebne podatke. Ko zberemo podatke, jih lahko obdelujemo glede na potrebe uporabnikov. To pomeni, da podatke glede na njihove zahteve pretvarjamo, razčlenjujemo in zgoščamo. Zbrane ali obdelane podatke nato shranjujemo za kasnejšo uporabo, lahko pa jih takoj posredujemo uporabnikom za njihove potrebe upravljanja, odločanja in nadzora.

2.1.3 Sestavine informacijskega sistema

Da bi informacijski sistem uspešno upravljal omenjene funkcije in uresničeval omenjene cilje, mora imeti določeno strukturo. Ta, splošno gledano, pomeni sintezo potrebnih elementov (Srića et al., 1995, str. 20–21).

Različni avtorji podajajo različne sestavine informacijskega sistema, vendar gre pri tem največkrat za različna poimenovanja ali različno členitev posameznih sestavin.

Glavne sestavine vsakega informacijskega sistema predstavljajo (Gradišar, Jaklič, & Turk, 2007, str. 40):

- strojna oprema,
- programska oprema,
- podatki,
- postopki in
- ljudje.

Strojna oprema je fizični del informacijskega sistema. Obsega vse fizične naprave, pripomočke ipd., ki jih uporabljamo v informacijskem procesu, da lahko zajema podatke in informacije ter omogoča njihovo povezovanje v skladno, funkcionalno, učinkovito ter uspešno celoto (Lesjak, 2003, str. 21).

Programska oprema obsega vse nize navodil za aktivnosti informacijskega procesa. Mednje ne štejemo le navodil, ki usmerjajo in nadzirajo računalnik (programi), temveč tudi navodila, ki jih potrebujejo ljudje (Lesjak, 2003, str. 21).

Podatki vstopajo v informacijski sistem, ta pa jih nato preoblikuje v informacije, ki predstavljajo glavni razlog obstoja informacijskega sistema (Lesjak, 2003, str. 21).

Postopki so povezovanja ter usklajevanja sestavin informacijskega sistema v skladno, funkcionalno, učinkovito ter uspešno celoto (Lesjak, 2003, str. 21).

Ljudje so potrebni pri izvajanju aktivnosti informacijskih procesov. Ta sestavina obsega informacijske in računalniške strokovnjake na eni strani ter uporabnike na drugi. Informacijski in računalniški strokovnjaki so ljudje, ki razvijajo in izvajajo informacijske procese. Uporabniki so ljudje, ki uporabljajo informacijski sistem oziroma informacije, ki jih ta zagotavlja (Lesjak, 2003, str. 21). Uporabniki informacijskega sistema so lahko notranji ali zunanji (Gradišar et al., 2007, str. 40).

2.2 Vrste informacijskih sistemov

Glede na vrsto problemov v vsebinskem smislu, pri reševanju katerih sodelujejo informacijski sistemi, večino teh razvrstimo v enega izmed standardnih tipov. Glede na to, v kolikšni meri informacijski sistemi prispevajo k izboljšanju strukture problemov, jih delimo v tri kategorije. To so informacijski sistemi, ki (Gradišar & Resinovič, 2001, str. 365):

- omogočajo dostop do orodij in informacij,
- pomagajo pri izvajanju postopkov po določenih pravilih in
- z avtomatizacijo procesov nadomeščajo ljudi.

Informacijske sisteme lahko proučujemo tudi z vidika različnih ravni usklajevanja dela. Glede na to ločimo naslednje štiri ravni (Gradišar & Resinovič, 2001, str. 366–367).:

- osebne informacijske sisteme (namenjeni so posameznikom),
- skupinske informacijske sisteme (povezujejo delovno skupino),
- organizacijske informacijske sisteme (vsebujejo povezave na ravni organizacije) in
- medorganizacijske informacijske sisteme (računalniško izmenjavanje informacij med organizacijami)

Glede na delovno prakso (z vsebinskega vidika), ki jo informacijski sistemi podpirajo, lahko informacijske sisteme v organizacijah najbolj grobo razdelimo na podsisteme, kot je organizacija razdeljena na organizacijske enote (Gradišar & Resinovič, 2001, str. 367):

- računovodski informacijski sistem,
- finančni informacijski sistem,
- proizvodni informacijski sistem,
- kadrovski informacijski sistem in
- marketinški informacijski sistem.

Informacijske sisteme, ki podpirajo delovno prakso, lahko proučujemo tudi z vidika pristopov in metod, ki so večinoma standardne in jih srečujemo v vseh navedenih informacijskih podsistemih. Glede na to ločimo sedem osnovnih tipov informacijskih sistemov (Gradišar et al., 2007, str. 48; Gradišar & Resinovič, 2001, str. 367):

- Izvajalni informacijski sistem
  Izhaja iz potreb po izvajanju oziroma iz temeljnega procesa organizacije. Glavna značilnost takega sistema je, da avtomatizira poslovanje. Njegova naloga je, da zbira in hrani podatke o poslovnih dogodkih ter včasih tudi nadzoruje odločanje, ki je del poslovnih dogodkov. Odpoved izvajalnega informacijskega sistema ima za organizacijo zelo neugodne in usodne posledice, ker se ustavi izvajanje temeljnega procesa. Zato je zelo pomembno, da je ta del informacijskega sistema zanesljiv.
- Informacijski sistem za upravljanje
  Zagotavlja informacije, ki so potrebne za upravljanje organizacije, oziroma ima svoje izhodišče v potrebah po informacijah za načrtovanje, ustvarjanje in nadziranje pravih stvari ob pravem času in na pravem mestu. Podatki teh sistemov so sešteti in združeni iz več izvajalnih informacijskih sistemov ter tako tudi ne nujno zelo natančni. Rezultati, ki jih ta sistem zagotavlja, se uporabljajo le znotraj organizacije in omogočajo sprejemanje pravih odločitev, s katerimi organizacija bolj učinkovito deluje.

- Sistem za podporo odločanju
  Osnovna značilnost sistema za podporo odločanju je v tem, da prek komunikacije z uporabnikom skušajo povečati njegove mentalne sposobnosti, kot so učenje, ustvarjalnost in sistematičen razvoj odločitev. Tako kot informacijski sistem za upravljanje tudi ta vrsta sistema skrbi za zbiranje podatkov, ki so potrebni pri sprejemanju odločitev. Ti podatki pa zahtevajo bolj analitičen pristop in so tako bolj zahtevni za uporabo. V splošnem so namenjeni reševanju slabo strukturiranih problemov. Uporabnik je običajno en sam in to vrsto informacijskega sistema uporablja neobvezno po lastni presoji.
- Vodstveni informacijski sistem
  Vodstveni informacijski sistem je sistem, ki v obliki pogovornega dela omogoča prilagodljiv dostop do informacij za spremljanje operativnih rezultatov in splošnih pogojev poslovanja. Oblikovan je tako, da vodstvu posreduje katerekoli informacije, kadarkoli jih potrebuje, in v obliki, ki je najbolj uporabna. Namen tega tipa informacijskega sistema je predvsem v identifikaciji priložnosti za zagotavljanje večje konkurenčnosti z drugimi organizacijami.
- Ekspertni informacijski sistem
  Ekspertni sistemi podpirajo umsko delo strokovnjakov, ki se ukvarjajo z oblikovanjem, s postavljanjem diagnoz ali z obvladovanjem kompleksnih situacij, pri čemer je potrebno znanje strokovnjaka na ozkem, dobro definiranem področju. Te sisteme zaradi uporabe znanja in podpore odločanja imenujemo tudi inteligentni sistemi. Kljub oznaki ekspertni sistemi ti sistemi niso pravi eksperti, ker ne znajo razmišljati po zdravi pameti, ampak le vsebujejo nekaj znanja in izkušenj, ki jih ima človek ekspert.
- Sistem za avtomatizacijo pisarniškega dela
  Omogoča vsakodnevne komunikacije in izvajanje informacijskih procesov v pisarnah znotraj organizacije. Njegov glavni namen je odprava ročnega dela z vključevanjem številnih orodij, kot so urejevalniki besedil, programi za elektronsko pošto, preglednice, telefonski sistemi itd. Uporaba tega tipa sistema je individualna in neobvezna. Ti sistemi načeloma ne vplivajo na vsebino informacij.
- Sistemi za podporo dela v skupini
  Predstavljajo novo in včasih ne povsem opredeljeno kategorijo na trgu računalniških programskih izdelkov, ki pomagajo pri organizaciji dela skupine ljudi. Osnovni cilj teh sistemov je povečati učinkovitost skupinskega dela z uporabo računalniške in komunikacijske tehnologije.

2.3 Strateški informacijski sistem

Hiter razvoj informacijske tehnologije prinaša organizacijam nove naloge, izzive in priložnosti. Sodobni informacijski sistemi so odprti in usmerjeni v podporo celotnemu spektru poslovnih funkcij, kajti samo tako lahko podjetju zagotavljajo konkurenčno prednost. Pri tem se bistveno poveča tudi kompleksnost informacijskih sistemov, zato

morajo podjetja veliko vlagati v informatiko, kljub temu pa se veliko projektov konča neuspešno. Organizacije kljub ogromnim vložkom na področju informacijske tehnologije ne izkazujejo ustreznih rezultatov v obliki povečanja produktivnosti in konkurenčnosti. K načrtovanju in izvajanju informacijskih rešitev je zato treba pristopiti strateško. Izdelati je treba strateški načrt informatike, ki predstavlja osnovni planski dokument razvoja informatike posamezne organizacije. Informacijsko funkcijo obenem umešča v skupni strateški načrt organizacije.

Trend sodobnega načina poslovanja narekuje vpeljavo strateških informacijskih sistemov, njihovega skrbnega načrtovanja in zagotavljanje skladnosti z izhodišči celovitega strateškega načrtovanja (Ward, 2002, str. 64).

Bobek (2007, str. 1) strateški informacijski sistem opredeli kot vsak informacijski sistem, ki v veliki meri vpliva na boljše poslovanje – poslovno uspešnost in konkurenčnost podjetja ter je osrednji dejavniki za uresničevanje poslovnih strategij v informatiziranih podjetjih.

Podobno meni tudi Gradišar (v Gradišar & Resinovič, 2001, str. 338), ki ugotavlja, da mora sistem, ki ima značilnosti strateškega informacijskega sistema, zadoščati dvema kriterijema:

- sistem je neposredno povezan in usklajen s poslovno strategijo,
- sistem pomembno vpliva na učinkovitost podjetja.

Informacijski sistemi so lahko strateški z vidika več ravni (Bobek, 2007, str. 2):

- **na ravni posameznih panog** vplivajo na izdelke in storitve, tržišča, novo ekonomiko proizvodnje in dejavnike konkurence,
- **na ravni podjetja (organizacije)** vplivajo na ustvarjanje novih poslovnih priložnosti, poslovne procese in organiziranost,
- **na ravni posameznih poslov** v podjetju pa podpirajo različne strategije poslov (nizkih stroškov, diferenciacije, usmeritve v tržne niše ipd.)

Če se organizacije lotevajo investicij na osnovi sprotnih potreb in pri tem ne upoštevajo strategije podjetja, potem imajo opravka s sistemom, ki ni strateški. To lahko privede do negativnih posledic za organizacijo, ki se kažejo kot (Ward, 2002, str. 47):

- investiranje v sisteme, ki ne podpirajo poslovnih usmeritev,
- sistemi niso integrirani (podvojitev naporov in podatkov),
- ni sredstva za določitev prioritet projektom, načrti se pogosto spreminjajo,
- slabo upravljanje informacij: niso dostopne, so nekonsistentne, netočne ali prepozne,

- nezadostne infrastrukturne investicije,
- vsi projekti so ovrednoteni le na finančni osnovi,
- problemi glede investicij v informatiko povzročajo konflikte med različnimi oddelki znotraj organizacije,
- nerazumevanje med uporabniki in informatiki vodi v konflikte in nezadovoljstvo,
- sistemi imajo večinoma krajšo življenjsko dobo, pogosteje je potreben ponoven razvoj, kar povzroča večje stroške.

Da se izognemo takim negativnim posledicam, mora strateško načrtovanje informacijskega sistema izhajati iz poslovnega načrta podjetja. Rezultat takega načrtovanja je strateški informacijski sistem. Ta organizaciji omogoči uresničitev ciljev in s tem posredno zagotovitev konkurenčne prednosti.

2.4 Možnosti pri zagotavljanju informacijskega sistema

V obdobju ideje novega informacijskega sistema je najpomembnejša odločitev o načinu izvedbe projekta. Pri tem se lahko odločamo med naslednjimi možnimi pristopi (Avison & Fitzgerald, 2002):

- lasten (notranji) razvoj, kjer podjetje samo izdeluje, gradi in razvija informacijski sistem,
- zunanje izvajanje dejavnosti informatike, kjer storitve informatike damo v izvajanje zunanjemu, specializiranemu izvajalcu,
- nakup programske rešitev, kjer podjetje informacijski sistem kupi pri zunanjem ponudniku.

Pri lastnem razvoju so prednosti predvsem, da člani projektne skupine dobro poznajo poslovno okolje, komunikacija v projektni skupini je hitra in stroški razvoja so običajno nižji. Slabosti pa predstavljajo pristranskost do problema, težje projektno vodenje in razporejanje človeških virov ter tako daljši čas razvoja.

Pri zunanjem izvajanju so glavne prednosti nepristranskost, zunanji izvajalec je specializiran za izvajanje dejavnosti informatike, stroški vlaganja v področje se porazdelijo med več strank, zunanji izvajalec lahko prinaša tudi izkušnje, saj dela za več podjetij, in stroški izvajanja storitev so jasni in vnaprej znani. Na drugi strani pa se slabosti kažejo predvsem v obliki nerazumevanja delovanja organizacije, višjih stroških ter možnosti razkritja poslovnih skrivnosti podjetja.

Predvsem v večjih podjetjih in organizacijah se pojavlja dilema o nakupu ali lastnem razvoju programskih rešitev. V splošnem velja, da se z nakupom precej zniža raven tveganja in skrajša čas uvedbe sistema, slabosti pa se kažejo v visoki ceni nakupa in

omejenosti pri prilagajanju sistema. Velja pravilo, da je nakup informacijske rešitve ob normalnih tržnih pogojih upravičen, če pokriva vsaj 80 % informacijskih potreb obravnavanega področja. Pri normalnih pogojih pomeni poleg ustrezne cene tudi razpoložljivost ustreznih rešitev v izvorni obliki in pripravljenost ponudnika za sodelovanje pri uvedbi ter prilagajanju rešitve (Kovačič, 1999, str. 40).

3 RAZVOJ INFORMACIJSKIH SISTEMOV

3.1 Težave izvedbe projektov razvoja informacijskih sistemov

Čeprav je informacijski sistem v zadnjih letih eden od najpomembnejših dejavnikov pri zagotavljanju konkurenčne prednosti organizacije, se na področju razvoja informacijskih sistemov srečujemo s številnimi težavami. Številne neodvisne raziskave kažejo, da imajo podjetja velike probleme z uspešnim dokončevanjem informacijskih tehnoloških projektov.

Ena izmed raziskav kaže, da je uspešnih projektov, ki so se zaključili v okviru načrtovanega časa, funkcionalnosti ter stroška, samo 32 %. Kar 44 % projektov, ki so bili udeleženi v raziskavi, je takih, ki so se sicer zaključili, vendar vsi elementi razvoja niso bili takšni, kot so bili dogovorjeni (prekoračen čas izvedbe, prekoračen proračun, premalo funkcionalnosti). Preostalih 24 % projektov raziskave je bilo prekinjenih ter odpovedanih že v času razvojnega cikla (The Standish Group International, 2009).

Attarzadeh (v Attarzadeh & Hock, 2008, str. 236) povzema glavne vzroke visokega odstotka neuspešnosti izvedbe projektov s področja informacijske tehnologije (v nadaljevanju IT):

- nepopolne zahteve in specifikacije,
- pomanjkanje sodelovanja z uporabniki,
- pomanjkanje sredstev,
- nerealistično postavljanje rokov,
- pomanjkanje podpore vodstva,
- spreminjajoče se zahteve in specifikacije,
- pomanjkanje načrtovanja,
- sistem ni več potreben,
- pomanjkanje pristojnosti IT-vodstva,
- pomanjkanje tehnoloških znanj in
- nepreizkušena tehnologija.

Standish Group (v Kappe, 2009) predstavi vzroke za konstantne neuspehe projektov razvoja informacijskih sistemov s pomočjo petih smrtnih grehov projektnega vodenja:

- **ambicija**, ki se lahko kaže v preveliki želji, da zgradimo nekaj prehitro, s preveč ljudmi ali v prevelikem obsegu funkcionalnosti,
- **aroganca,** do katere lahko pride s strani vodje projektov, ki preglasi uporabnika in izsili implementacijo po svojih zmotnih prepričanjih,
- **ignoranca** v primeru, ko želimo v novem sistemu implementirati vse funkcionalnosti starega, tudi če ne poznamo njihovih podrobnosti, kar lahko hitro pripelje do problemov,
- **goljufivost**, ko ljudje prikrivajo stroške ali časovno oceno projekta z namenom, da bi od vodstva dobili dovoljenje za zagon projekta,
- **abstinenca**, ko ključni ljudje niso vključeni v projekt.

Navedeni problemi potrjujejo dejstvo, da je razvoj informacijskih sistemov zelo kompleksno opravilo in se ga je treba lotiti premišljeno. Splošno sprejet pristop za reševanje zgoraj omenjenih problemov predvideva uvedbo sistematičnih smernic za razvoj informacijskih sistemov v okviru organizacije. Znanje, ki je za to potrebno, se nahaja v obliki metodologij razvoja informacijskih sistemov.

3.2 Metodologija razvoja informacijskih sistemov

3.2.1 Opredelitev osnovnih pojmov metodologije razvoja informacijskih sistemov

Metodologijo lahko splošno opredelimo kot skupek postopkov, tehnik, orodij in pripomočkov za dokumentiranje, ki koristijo razvijalcem sistema pri njihovem prizadevanju implementirati nov informacijski sistem. Metodologijo sestavljajo faze, ki so sestavljene iz podfaz in vodijo razvijalce rešitev pri izbiri tehnik, ki so primerne v fazi projekta, ter jim pomagajo načrtovati, upravljati, kontrolirati in ocenjevati projekte razvoja programskih rešitev (Avison & Fitzgerald, 2002, str. 20).

Avtorja nadalje ugotavljata, da je metodologija več kot le skupek njenih sestavin. Navadno je osnovana na nekem filozofskem pogledu, sicer se ne razlikuje od metode. Metodologije se lahko razlikujejo v posameznih tehnikah, ki jih priporočajo, v postopkih, ki jih predpisujejo, ali v vsebini posamezne faze, toda včasih gre celo za bolj temeljne razlike.

Metodologija predstavlja znanje o vseh oblikah in načinih raziskovanja, s katerimi je mogoče doseči objektivno in sistemsko znanje. Tako pod pojmom »metodologija razvoja informacijskega sistema« razumemo celoto načel, konceptov, pravil, metod in tehnik, ki se uporabljajo za dosego ciljev projektiranja, gradnje in vzdrževanja informacijskega sistema (Srića et al., 1995, str. 259).

Cockburn (2001, str. 101) metodologijo deli na 13 osnovnih pojmov:

- vloge: ki jih prevzemajo in opravljajo zaposleni ter določajo njihovo delo,
- skupina: je združba ljudi z različnimi vlogami, ki deluje na določenem projektu,
- postopki: predstavljajo osnovo metodologije, ki določa postopke, ki jih je treba izpeljati za dosego ciljev,
- aktivnosti: predstavljajo dela, ki jih zaposleni izvajajo,
- orodja: so pripomočki, ki jih uporabljamo pri razvoju. Njihov namen je omogočiti čim hitrejše delo,
- izdelek: je to, kar poskušamo ustvariti s pomočjo metodologije,
- kakovost: določa natančnosti izdelave pri aktivnostih,
- standardi: vplivajo na delovanje in videz izdelka,
- proces: je točno določeno zaporedje izvajanja aktivnosti,
- mejniki: z njimi določamo stanje projekta in pokažemo, kaj je v okviru projekta še treba narediti,
- strokovno znanje: je eden od osnovnih kriterijev opravljanja določene vloge,
- skupinske vrednote: so skupek splošno sprejetih vrednosti razvojne skupine, ki pomembno vplivajo na elemente metodologije,
- osebnost: je vzorec delovanja posameznika.

Različne metodologije lahko sledijo različnim ciljem. Ti so lahko na primer (Avison & Fitzgerald, 2003, str 21):

- natančno zabeležiti zahteve informacijskega sistema. Metodologija naj bi pomagala uporabnikom in razvijalcem določiti, proučiti in analizirati uporabniške zahteve na način, da bo informacijski sistem lahko izpolnjeval potrebe uporabnikov,
- določiti sistematično metodo razvoja programske rešitve, kjer se lahko napredek učinkovito meri in opazuje. Dobro načrtovane faze razvoja in vmesni cilji v metodologiji razvoja omogočajo učinkovitost tehnik projektnega načrtovanja,
- razviti informacijski sistem v okviru časovnega roka ob sprejemljivih stroških. Čas, namenjen posameznim tehnikam v metodologiji, naj bo omejen, sicer lahko pri iskanju popolnosti izgubimo ogromno časa,
- razviti sistem, ki je dobro dokumentiran in enostaven za vzdrževanje. Izvedba vzdrževanja in nadaljnjega razvoja, nadgradenj ter sprememb sistema zahteva kakovostno dokumentacijo, ki jo predpiše izbrana metodologija,
- spremembe v procesu razvoja informacijskega sistema ugotoviti čim bolj zgodaj. Stroški, povezani z realizacijo sprememb, se naglo povečujejo v času razvoja informacijskega sistema. Prej ko spremembe realiziramo, manjši so stroški projekta,
- izdelati informacijski sistem, ki ustreza ljudem, na katere vpliva sistem. Če informacijski sistem ustreza naročnikom sistema, je večja verjetnost, da se bo informacijski sitem uporabljal in bo uspešen.

Čeprav so na tržišču na voljo različne formalne metodologije za razvoj informacijskega sistema, organizacije z njimi niso povsem zadovoljne. Formalnim metodologijam zato pogosto dodajo neformalne elemente, ki so posledica načel, izkušenj in znanja, ki ga člani organizacije uporabljajo pri svojem delu. Prilagojena metodologija tako vsebuje vrednote, ki so organizaciji lastne in so njena konkurenčna prednost.

3.2.2 Prednosti in slabosti uporabe metodologije razvoja informacijskih sistemov

Z uporabo metodologij pri razvoju informacijskih sistemov se pričakujejo nekatere prednosti. Fitzgerald podaja naslednje prednosti uporabe metodologije razvoja informacijskih sistemov (Fitzgerald, 1996, str. 6–8):

- razdelitev kompleksnega procesa v obvladljive naloge. Princip »deli in vladaj« se mnogokrat pojavlja v matematičnih in inženirskih pristopih reševanja kompleksnih problemov. Enak pristop uporabljajo tudi metodologije razvoja informacijskih sistemov, kjer celoten proces razvoja razdelijo na model več lažje obvladljivih enostavnih nalog,
- poenostavitev projektnega vodenja in nadzora. Metodologija omogoča skladno ogrodje, v katerem usmerja in priporoča tehnike, revizijske postopke, kakovost, dobre prakse ipd. ter tako olajšuje nadzor,
- priporoča ogrodje za izbiro pristopov in virov. Razvojna metodologija tudi priporoča strukturno ogrodje v razvojnem procesu z zagotavljanjem vrste potrebnih sestavin razvoja,
- zagotavlja ekonomičnost s specializacijo in z delitvijo dela. Z razdelitvijo celotnega procesa razvoja v serijo samostojnih faz je omogočena tudi delitev dela v posameznih fazah. Naloge so tako lahko dodeljene ljudem, ki imajo izkušnje in so specializirani za opravljanje posamezne naloge,
- omogoča sistemizacijo znanja in epistemološko utemeljitev. Metodologija predstavlja okvir za pridobivanje in sistemizacijo znanja. To nadalje omogoča prenos znanja z manj izkušenih na bolj izkušene udeležence v razvojnem procesu ter tako spodbuja ter zagotavlja učinkovit prenos znanja znotraj organizacije,
- standardizacija razvojnega procesa. Sledenje standardiziranemu razvojnemu procesu močno izboljša koordinacijo in komunikacijo med udeleženci znotraj procesa. Izboljša se tudi disciplina razvoja, kar pomeni večjo produktivnost in hitrejši razvoj,

Kljub prednostim uporabe metodologije razvoja informacijskih sistemov pa Fitzgerald nadalje podaja tudi njene slabosti (Fitzgerald, 1996, str. 12–19):

- opredelitvene anomalije. Odločanje, kaj dejansko sestavlja izbrano razvojno metodologijo ter čemu je določena metodologija namenjena, je lahko zelo problematično. Številne metodologije poleg pomembnih razlik, ki ločijo določeno

metodologijo od druge, vsebujejo tudi številne nepomembne in umetno ustvarjene razlike,

- posplošitev brez ustreznega konceptualnega in izkustvenega okvira. Metodologije naj bi bile zgrajene na osnovi metod in tehnik, uporabljenih v uspešnih razvojnih projektih. Kljub temu najbolj popularne metodologije temeljijo na intuicijah njihovih avtorjev, ki navadno delujejo povsem drugje kot v okviru realnih projektov,
- pomanjkljivost racionalnih znanstvenih vzorcev. Razvoj sistemov ni le zaporedni tehnični proces izvajanja enostavnih nalog, kot ga opredeljuje metodologija. Upoštevati mora tudi številne nepredvidljive dejavnike,
- zamenjava cilja. Do zamenjave cilja pride v razvojnem procesu, ko udeleženci slepo sledijo in se ukvarjajo s sledenjem priporočil metodologije, pri tem pa pozabljajo na cilj procesa. Ta je razvoj informacijskega sistema,
- predpostavka, da je metodologija splošen pristop. Domneva, da se lahko uporabi enotno metodologijo za vse vrste projektov v vseh organizacijah, ne drži. Zavedati se je treba, da je metodologija le pomoč pri organizaciji in uokvirjanju problemov,
- neustrezno prepoznavanje dejavnikov udeležencev. Metodologija predpisuje udeležence in njihove vloge. Pri tem ne more upoštevati dejanskih sposobnosti posameznih udeležencev razvojnega procesa. Zavedati se je treba, da razvoj izvajajo ljudje in ne metodologija sama. Ta le predstavlja uporaben okvir udeležencev organizacije.

3.2.3 Teža metodologije razvoja informacijskih sistemov

Ob koncu 90. let so se kot kritika obsežnih formalnih metodologij začeli uveljavljati novi pristopi pri razvoju informacijskih sistemov. Kot posledica tega se je je pojavila tudi delitev metodologij na težke in lahke metodologije.

Teža metodologije je opredeljena kot produkt njenega obsega in gostote (Cockburn, 2002, str. 107):

Teža metodologije = obseg metodologije x gostota metodologije

- obseg metodologije je določen s številom različnih elementov, ki jih metodologija opisuje. Obseg opredeljuje, katere osnovne in podporne postopke neka metodologija obsega, katere vloge so v njej vsebovane, katere standarde upošteva ipd.,
- gostota metodologije je opredeljena kot zahtevana raven podrobnosti oziroma formaliziranosti opisa njenih elementov. Metodologije z višjo gostoto so bolj formalne, njihovi elementi pa opisani bolj podrobno. Metodologije z nižjo gostoto prepuščajo več stvari interpretaciji posameznika, ki metodologijo uporablja.

3.3 Klasične metodologije

Klasične metodologije se v praksi uporabljajo že zelo dolgo časa. Nastale so na začetku snovanja znanosti področja razvoja informacijskih sistemov kot posledica potrebe po nadzoru razvojnega procesa. Običajno jih štejemo v skupino težkih metodologij, saj vsebujejo predstavnike z večjo gostoto in večjim obsegom. Njihova glavna značilnost je zaporedni, discipliniran in v celoti predvidljiv pristop prek vseh faz razvojnega procesa. Glavni poudarek dajejo analizi in načrtovanju, ki podrobno opisuje projekt za dolgo časovno obdobje, ter obravnavajo dokumentacijo kot ključni del projekta. Drži se jih sloves, da so zelo birokratske, kar upočasnjuje hitrost razvoja. Vse temeljijo na sistematičnih metodah, kjer se faza gradnje ne pojavi pred zaključkom natančno opredeljene faze načrtovanja. Klasične metodologije sprejemajo vse glavne odločitve projekta v fazi načrtovanja. Faza gradnje le sledi rezultatom faze načrtovanja in je tako popolnoma predvidljiva. Prav te značilnosti klasičnih metodologij podajajo razlog njihove uspešne ali neuspešne uporabe.

Stoimen (2011) podaja splošne značilnosti tradicionalnega načina razvoja informacijskih sistemov:

- primeren je za uporabo pri zelo obsežnih projektih z večjimi razvojnimi skupinami,
- faza načrtovanja je zelo natančna,
- dokumentacija je obsežna in natančno opisuje vse funkcionalnosti sistema,
- primeren je za projekte, kjer so zahteve definirane v zelo zgodnji fazi razvoja in se v času razvojnega procesa ne spreminjajo,
- vse faze razvojnega procesa si sledijo zaporedno.

V nadaljevanju predstavljam najpomembnejše predstavnike modelov in pristopov klasičnih metodologij ter podajam njihove glavne značilnosti.

3.3.1 Modela »od zgoraj navzdol« in »od spodaj navzgor«

»Od zgoraj navzdol« ter »od spodaj navzgor« sta pristopa, ki se uporabljata v procesu razvoja kot dve vrsti razvojnih metodologij. Oba pristopa se lahko šteje kot nadgradnjo obstoječih razvojnih metodologij.

Pristop »od zgoraj navzdol« se je pojavil v 70. letih prejšnjega stoletja. Predvideva pregled zgradbe sistema brez poudarka na podrobnostih katerega izmed njegovih delov. Vsak posamezni del sistema se kasneje razvija naprej z načrtovanjem več podrobnosti. Vsak nov del sistema se lahko ponovno izboljša in bolj podrobno določi, dokler celotna specifikacija sistema ni toliko podrobna, da se lahko prične z gradnjo. Pristop »od zgoraj navzdol« daje poudarek načrtovanju in popolnemu razumevanju sistema. Zanj je značilno, da se z gradnjo

ne prične toliko časa, dokler ni dosežena dovolj podrobna raven vsaj na nekaterih delih sistema.

Nasprotno pristop »od spodaj navzdol« posamezne dele sistema podrobno določi že na začetku. Tako se jih lahko tudi že implementira. Posamezne zgrajene dele se nato združuje skupaj v večje dele sistema, ki se jih ponovno medsebojno povezuje, dokler ni zgrajen celoten sistem. Pristop ima poudarek na gradnji, s katero se lahko prične takoj, ko je določen prvi del sistema. Ker posamezni deli v fazi gradnje nimajo jasne ideje, kako se bodo povezali z ostalimi deli sistema, s tem nosijo tveganje, da povezovanje ne bo tako preprosto, kot je običajno videti na prvi pogled.

Moderni pristopi razvoja sistemov običajno kombinirajo metode obeh navedenih pristopov. Čeprav je dobro razumevanje celotnega sistema običajno del dobrega načrta, obstaja veliko projektov, kjer se poskuša uporabiti že izdelane gradnike. S tem se tudi usmeri načrtovanje, ki je v takem primeru »od spodaj navzgor«.

3.3.2 Zaporedni ali slapovni model

Zaporedni ali slapovni model je metodologija razvoja ki jo je konec 60. let prejšnjega stoletja predlagal National Computing Centre. Temelji na empiričnih ugotovitvah, da cena spremembe raste eksponentno s stopnjo sprememb. Zaključek je, da je pomembne odločitve treba sprejeti kar se da na začetku, ker so kasnejše spremembe drage. Proces se imenuje tudi slapovni, ker vsaka faza v modelu logično sledi predhodni fazi. Naslednja faza se začne, ko se predhodna konča (Avison, 2003, str. 31).

V literaturi zasledimo nekaj različnih zaporednih modelov, ki se razlikujejo predvsem po podrobnosti delitev posameznih faz. Avison navaja, da v zaporednem modelu razvoj linearno poteka po naslednjih fazah (Avison, 2003, str. 27):

- študija izvedljivosti,
- raziskovanje sistema,
- analiza sistema,
- načrtovanje,
- izvedba in
- pregled ter vzdrževanje.

Študija izvedljivosti je namenjena podrobnejši proučitvi izvedljivosti projekta. Za vsak projekt je treba proučiti izvedljivost z naslednjih vidikov (Avison, 2003, str. 28):

- legalnosti: ali je sistem skladen z nacionalnim in organizacijskim zakoni,
- organizacijskega in socialnega: ali je sistem sprejemljiv za organizacijo ter njene zaposlene,

- tehničnega: ali sistem lahko podpremo z obstoječo tehnologijo ter ali obstaja dovolj strokovnjakov za njegovo izgradnjo,
- ekonomskega: ali je finančno dosegljiv in stroškovno opravičen.

Ko vodstvo odobri nadaljevanje projekta, sledi faza raziskovanja sistema. V njej se poda odgovore na (Avison, 2003, str. 28):

- funkcionalne zahteve obstoječega sistema,
- zahteve novega sistema,
- omejitve,
- območja podatkovnih tipov in količino podatkov,
- izjeme,
- probleme sedanjih metod dela.

Na podlagi zbranih dejstev preidemo v fazo analize sistema, kjer se do podrobnosti poskuša razumeti vse vidike trenutnega sistema, razloge za uporabo določenih metod, obstoj alternativnih metod ter podati, kako se stvari lahko izboljša v novem sistemu. Poskušamo razumeti razloge za trenutno stanje in nakazati načine za njegovo izboljšanje.

Naloga faze načrtovanja je izdelati načrt na podlagi modelov in ostalih specifikacij, ki so bile pripravljene v fazi analize. V ospredje stopi vprašanje, kako izpolniti v predhodni fazi dogovorjene zahteve. V sklopu te faze je treba še do konca izdelati dokumentacijo, ki naj bi vsebovala (Avison, 2003, str. 30):

- vhodne podatke in njihov zajem,
- izhode sistema,
- procese za transformacijo vhodnih podatkov v izhodne,
- podatkovne strukture (zapise), ki se pojavljajo znotraj sistema,
- varnostna določila ter restavriranje in
- izvedbeni načrt.

Sledi faza izvedbe, kjer se razvije sistem glede na načrt, izdelan v fazi načrtovanja. V začetni fazi vsebuje dela, kot so nakup in instalacija opreme, kodiranje ter testiranje programov in oblikovanje dokumentacije. Nadalje je treba izvesti integralni preizkus sistema tudi s strani uporabnikov ob uporabi večje količine podatkov. Izvesti je treba tudi šolanje uporabnikov sistema in izvesti prehod s starega na novi sistem. Po končani fazi je sistem vpeljan in operativen.

Po uvedbi je na novem sistemu treba spremljati njegovo delovanje, se odzvati na odkrite probleme v tem času, izvajati nadgradnje in odpravljati napake ter načrtovati dodelave. Vse

to se izvaja v zadnji fazi življenjskega cikla zaporednega modela – fazi izvajanja in vzdrževanja.

Glavne prednosti modela so preglednost, prepoznavnost in dobra organiziranost. Za izvajanje faz morajo imeti dobro opredeljene zahteve, zato je ta model razvoja usmerjen v izdelavo obsežne dokumentacije. Glavna pomanjkljivost modela je toga razdelitev projekta na točno določene zaporedne faze, zaradi česar se slabo odziva na spreminjajoče se zahteve naročnika ter v takšnih primerih zahteva težak ter drag povratek v predhodno fazo. Pri zaporednem modelu je rezultat dela viden šele na koncu razvojnega cikla. Bistvene pomanjkljivosti se tako lahko pokažejo šele na koncu razvoja, ko vsaka sprememba pomeni nov velik strošek za naročnika.

V današnjem času zaporedni model hitro izpodrivajo metodologije, ki vsebujejo bolj agilne pristope. Glavna razloga sta težak in drag povratek v predhodno fazo ter dolgo trajanje projekta. Kljub temu je zaporedni model ali ena od njegovih izpeljank še vedno zelo pogosto uporabljen v praksi (CA Technologies survey, 2010; West, 2010).

Reševanje navedenih problemov zaporednega modela razvoja je pripeljalo do razvoja novih in drugačnih modelov življenjskega cikla sistema.

3.3.3 Spiralni model

Spiralni model je pristop oziroma metodologija razvoja sistema, ki združuje elementa načrtovanja in izdelave prototipov v fazah. Je svojevrstna mešanica pristopov »od zgoraj navzdol« in »od spodaj navzgor«. Model je razvil Boehm (1986) na podlagi izkušenj z različnimi izboljšavami zaporednega modela za uporabo razvoja pri velikih projektih. Največja značilnost modela je neposredno prepoznavanje in odstranjevanje tveganj.

Grafični prikaz modela je oblika spirale v koordinatnem sistemu. Vsaka zanka spirale predstavlja eno fazo razvojnega procesa. Obstajajo štiri glavne faze spiralnega modela (Boehm, 1988, str. 3):

- določitev ciljev – opredeljeni so razumljivi cilji projekta,
- načrtovanje sistema – napravi se načrt razvoja sistema,
- razvoj v korakih – postopoma se razvije celi sistem,
- ovrednotenje in testiranje – ustrezen model je izbran in testiran.

Prednosti spiralnega modela so:

- neposredno prepoznavanje tveganj, kar omogoča njihovo učinkovito odstranitev,
- ocena stroškov in načrt sta z napredovanjem projekta bolj realistična, ker še povečuje število vprašanj (podajanje informacij iz prejšnjih faz),

- omogoča lažje obvladovanje sprememb,
- omogoča hitrejši pričetek s fazo izvedbe projekta.

Glavna pomanjkljivost modela je zelo groba začetna ocena stroškov in plana, ker nekatere analize niso zaključene, vse dokler zanka ne preide faze načrtovanja.

V današnjem času se model zelo malo uporablja (Forrester Research v West, 2010, str. 2). Kljub temu je močno vplival na moderne agilne pristope razvoja sistemov, za katere velja, da so v pristopu veliko bolj ekstremne kot spiralni model.

3.3.4 Prototipni model

Prototipni model je proces razvoja informacijskih sistemov, ki se prične z zbiranjem zahtev, iz katerega sledita izdelava prototipa in njegova evolucija s strani uporabnikov (naročnika). Uporaba prototipov naslavlja nekatere probleme zaporednega pristopa ter še posebej povečuje udeležbo uporabnikov v procesu nastajanja sistema. Razvoj temelji na izdelavi prototipa, ki je nepopolna izdelana različica z osnovnimi funkcionalnostmi želenega sistema. To omogoča lažje komuniciranje med razvijalci in naročnikom, ki po navadi ni tehnično podkovan in tako lažje ter natančneje opredeli zahteve za končno rešitev.

Uporaba prototipov tako izboljšuje faze raziskave, analize in načrtovanja sistema. Še posebej je uporabna v naslednjih primerih, ko (Avison & Fitzgerald, 2003, str. 90–91):

- področje rešitve ni dovolj dobro definirano,
- organizacija ne pozna dobro nove tehnologije, ki je potrebna za izdelavo rešitve,
- komunikacija med analitiki in uporabniki ni dovolj dobra,
- strošek zavrnitve rešitve s strani uporabnikov bi bil visok in je zagotovitev pokritja uporabniških potreb bistvenega pomena,
- potrebna je ocena vpliva prihodnjega informacijskega sistema.

Pri tem modelu razvoja ločimo dva različna pristopa (Avison & Fitzgerald, 2003, str. 90):

- evolucijski prototip, kjer prek posameznih faz izboljšujemo prototip, dokler ne pokrijemo vseh funkcionalnosti, in
- prototip, ki ga po uporabi zavržemo, ker služi le kot predstavitev določenih funkcionalnosti ali kot pomoč pri simulaciji ključnega dela sistema.

Kadar je prototip namenjen kasnejši operativni uporabi, je izjemnega pomena kontrola nad procesom razvoja, da se izognemo iskanju bližnjic, ki neizbežno povzročijo probleme pri nadaljnji uporabi tako nastale rešitve.

3.3.5 Model RAD

Model hitrega razvoja programskih rešitev ali Rapid application development (RAD) je uvedel Martin (1991) kot odgovor na probleme zaporedne metodologije razvoja informacijskih sistemov, pri katerih so se lahko zaradi dolgotrajnega razvoja zahteve spremenile, še preden je bil sistem v celoti razvit, kar je pogosto vodilo v njegovo neuporabnost. Ključni dejavnik hitrega razvoja je izdelava delujočih poslovnih aplikacij v krajšem časovnem okviru z manjšo investicijo. Sistem se razvije po funkcionalnih delih v pospešenih in časovno omejenih ciklih. Uporabniki in razvijalci so običajno vključeni na delavnicah, kjer skupaj oblikujejo zahteve sistema. Za metodologijo je značilno, da različne vidike sistema lahko razvije ob različnem času, pri čemer je najpomembnejši prvi časovni okvir. Celoten sistem je tako razvit postopno prek več ciklov ob sodelovanju majhnih timov uporabnikov in razvijalcev.

RAD-metodologijo določajo naslednje značilnosti (Avison & Fitzherald, 2003, str. 94–99):

- **iterativni in inkrementalni razvoj:** uporabniki zelo težko z gotovostjo podajo vse svoje zahteve že na začetku snovanja sistema, zato sta potrebna iterativni in inkrementalni pristop pri reševanju te problematike. RAD je proces organizacijskega učenja prek iteracij načrtovanja, grajenja, testiranja in razvijanja. Zelo težko je uporabniku navesti tudi vse podrobnosti zahtev, zato RAD uporablja pristop, kjer se zahteve navedejo do stopnje, ki je primerna za trenutno fazo razvoja. Vse podrobnosti se poda šele kasneje, ko projekt napreduje prek inkrementalnega razvoja in pride do dejanske izdelave funkcionalnosti,
- časovni okviri: razvoj celotnega sistema se razdeli v večje število komponent oziroma časovnih okvirov, ki določajo njihovo ločeno gradnjo. Najprej se razvije najbolj bistvene funkcionalnosti sistema, ki so čim hitreje dane na razpolago v prvem časovnem okviru. Časovni okvir naj ne bi bil daljši od 90 dni,
- **Paretovo pravilo:** prepričanje zagovornikov RAD je, da okoli 80 % funkcionalnosti sistema lahko dosežemo z okoli 20 % naporom, potrebnim za izvedbo celotnih zahtev sistema. To pomeni, da 20 % zahtev, ki po navadi zahtevajo največ napora ter časa, izvajamo na koncu.

Pravila MOSCOW. V metodologiji RAD funkcionalnosti projekta razdelimo na:

- funkcije, ki jih je nujno implementirati (angl. *the Must Haves*),
- funkcije, ki bi jih bilo dobro implementirati (angl. *the Should haves*),
- funkcije, ki bi jih lahko implementirali (angl. *the Could Haves*) in
- funkcije, ki jih ne bomo implementirali (angl. *the Won't Haves*).

Časovni okviri nato obvezno vključujejo vse funkcionalnosti prve kategorije (angl. *the Must Haves*), vsaj nekaj funkcionalnosti druge kategorije (angl. *the Should haves*) in kakšno funkcionalnost tretje kategorije (angl. *the Could Haves*).

JAD-delavnice. Te učinkovito zamenjujejo tradicionalne načine opredeljevanja zahtev, kjer analitik opravlja intervjuje s posameznimi nosilci funkcij in z uporabniki ter naknadno usklajuje nerešene zadeve, kar lahko traja zelo dolgo. Prisotnost pooblaščenih ljudi omogoča odločanje o projektu znotraj delavnice, s čimer je dosežena večja preglednost odločanja.

Prototipiranje. Je pomemben gradnik RAD-metodologije razvoja, saj omogoča pohitritev procesa zbiranja zahtev s tem, ko uporabniku poskuša predstaviti delovanje sistema v praksi. Prototipiranje je bolj natančno opredeljeno v poglavju 3.3.4.

Sponzor in prvak. Uspešna RAD-metodologija zahteva prisotnost sponzorja in prvaka sistema. Sponzor je oseba, ki je potrjena s strani naročnika sistema, ki je zavezan projektu ter učinkovito odstranjuje birokratske in politične ovire, ki bi lahko zadržale izvajanje projekta. Prvak je izkušena oseba, vključena v projekt, ki razume in verjame v metodologijo razvoja RAD ter je tako pripravljena voditi projekt naprej in pri tem uspešno odstranjevati birokratske ter politične ovire.

Orodja. Metodologija predvideva uporabo različnih orodij, ki zagotavljajo tehnično integriteto pri modeliranju in načrtovanju sistema. To nam omogoča hitro izdelavo ustrezne rešitve. Ključna je kakovost orodja, saj se integriteta projekta zagotavlja skozi uporabo skupnega podatkovnega in procesnega modela.

Model RAD znotraj posameznega časovnega okvira (cikla) vsebuje štiri faze, ki vključujejo dejavnosti in naloge, potrebne za opredelitev obsega in poslovnih zahtev sistema ter njegov razvoj, izvajanje in uporabo:

- načrtovanje zahtev – faza opredli poslovne funkcije in zahteve, ki jih bo rešitev vsebovala,
- funkcionalna zasnova – v tej fazi se razvije delujoči prototip programske rešitve, ki vsebuje modele procesov in zahtev končne rešitve,
- razvoj – ta faza zaključi razvoj rešitve s pomočjo izboljšave prototipa. Tega uporabniki preizkušajo ter vračajo razvojnemu timu za popravilo ali predelavo toliko časa, dokler se prototip ne razvije v končno rešitev,
- uvajanje – faza vključuje preizkušanje in usposabljanje končnega uporabnika za pravilno uporabo rešitve.

RAD-metodologija je postala zelo priljubljena metodologija predvsem zaradi hitrega razvoja rešitev ter visoke stopnje ustreznosti rešitve končnemu uporabniku ob nizkih

stroških izdelave ter vzdrževanja rešitve. Kljub temu je metodologija deležna več kritik glede iskanja bližnjic na račun hitrosti izdelave predvsem z vidika kakovosti in dokumentiranosti rešitve (Avison & Fitzgerald, 2003, str. 99).

3.4 Agilne metodologije

Največja kritika klasičnih metodologij je njihova birokratska naravnanost. Obsegajo veliko obsežnih nalog, ki jih je treba opraviti po klasični metodologiji, kar močno upočasnjuje razvoj rešitve. Agilne metode razvoja informacijskih sistemov poskušajo ponuditi odgovor na iskanje manj obsežnih metodologij razvoja, ki s seboj prinesejo hitrejše procese razvoja. Agilno pomeni pripravljeno na gibanje, aktivnost, živahnost in hitrost (Abrahamsson, 2002).

Agilni razvoj ni določen zaporedni proces, ki mu sledimo. Je filozofija in razmišljanje o razvoju rešitev. Predstavlja skupino različnih metod in pristopov, ki temeljijo na podobnih načelih. Na splošno velja, da agilne metode spodbujajo proces vodenja projekta, ki vsebuje testiranje in prilagajanje, timsko delo, organizacijo, če je to potrebno ter po lastni odgovornosti članov, razvoj v celoti prilagojen potrebam uporabnika in ciljem ter uporabo najboljših praks, ki omogočajo hitro gradnjo rešitev visoke kakovostni (Shore, 2007, str. 9).

Fowler (2003) meni, da obstajata dve ključni značilnosti agilnih metodologij:

- agilne metodologije razvoja informacijskih sistemov se raje prilagajajo kot predvidevajo. V klasičnem pristopu razvoja rešitev je večji del razvoja načrtovan do podrobnosti za daljše časovno obdobje razvoja rešitve. To deluje, dokler se zahteve in okolje ne spreminjajo. Agilne metodologije razvoja pa sprejemajo spremembe, jim prilagajajo proces razvoja rešitve in se tudi same spreminjajo,
- agilne metodologije razvoja informacijskih sistemov so bolj usmerjene k ljudem kot k procesom razvoja rešitev. Ideja metodologij, ki so procesno usmerjene, je, da vsebujejo definiran proces, ki deluje ne glede na to, kdo ga izvaja. Agilne metodologije privzemajo, da proces ne more nadomestiti veščin razvojne skupine. Proces naj le podpira razvijalce pri njihovem delu.

Do pojma agilnega razvoja je prišlo sredi 90. let prejšnjega stoletja kot odgovor na uporabo težkih metodologij, kot je zaporedni model. Zaradi tega so se agilne metodologije na začetku imenovale tudi lahke metode. Leta 2001 je ustanovljena neprofitna organizacija Agile Allianze (Agilna zveza), ki promovira agilni razvoj. V naslednjih mesecih so izoblikovali seznam skupnih vrednot in principov, ki naj bi razvijalcem omogočal hitro in učinkovito delo z visoko odzivnostjo na spremembe. Rezultat dela je manifest agilnega razvoja, ki je sestavljen iz štirih osnovnih vrednot ter 12 bolj podrobnih napotkov.

Glavne vrednote agilnega pristopa, objavljene v manifestu, so (Manifesto for Agile Software Development, 2001):

- posameznik in interakcije med procesi ter orodji: agilni napredek poudarja razvijalca ter človeško vlogo v nasprotju z institucionalnimi postopki in razvojnimi orodji. S poudarkom na komunikaciji in povezanosti med razvijalci se izboljšujejo delovno okolje, motivacija in občutek pripadnosti. Če za projektom ne stojijo sposobni ljudje, tudi odličen razvojni proces, projekt ne bo uspešen. Sposoben razvijalec ni samo strokovnjak na svojem področju, razvite mora imeti tudi komunikacijske sposobnosti, sposobnosti timskega dela in mora biti prilagodljiv spremembam v okolju,
- delujoča rešitev pred obsežno dokumentacijo: ključna naloga razvojne ekipe je zagotavljanje nove delujoče in preverjane rešitve, ki se izdaja v rednih intervalih. Razvijalci morajo kodo ohraniti enostavno in kolikor je mogoče dovršeno. Poudarek ni na količini nastale dokumentacije,
- sodelovanje z naročnikom pred pogajanji na osnovi pogodbe: odnos in sodelovanje med razvijalci ter naročnikom ima prednost pred strogo pogodbo. Pogodba je osnova, vendar je v njej poudarjena komunikacija med obema stranema. S poslovnega vidika je agilni razvoj osredotočen na hitro dostavo, s čimer se zmanjšajo številne nevarnosti (sprememba zahtev, testiranje, odstop od projekta in druge),
- odzivnost na spremembe pred sledenjem načrtu: razvojna skupina, ki vključuje razvijalce ter naročnika, mora biti v času razvojne poti projekta dobro obveščena, verodostojna in pristojna, da upošteva številne prilagoditve. To pomeni, da morajo udeleženci biti ves čas pripravljeni na spremembe.

Cockburn (2001, str. 149) definira jedro agilnih metodologij razvoja z uporabo neobsežnih, vendar zadostnih v človeško komunikacijsko usmerjenih pravil ravnanja v projektu, s katerimi agilni proces postane prav tako enostaven in zadosten. Ta pravila so:

- dva do osem ljudi v enem prostoru, kar omogoča boljše komuniciranje in izmenjavo znanja,
- pogosta uporaba strokovnjakov, kar omogoča kratke in stalne povratne informacije,
- kratke inkrementalne različice, ki naj trajajo od enega do treh mesecev. To omogoča hitro testiranje in odpravljanje napak,
- popolnoma avtomatizirano regresijsko testiranje (delno in funkcionalno testiranje), ki stabilizira kodo ter omogoča njeno nenehno izboljševanje,
- izkušeni razvijalci, ki pospešujejo čas razvoja od 2- do 10-krat v primerjavi s počasnejšimi člani ekipe.

Miller podaja naslednje značilnosti agilnega procesa razvoja z vidika hitre dostave, kar omogoča skrajšanje življenjskega cikla projekta (Miller, 2001):

- modularnost stopenj razvojnega procesa,
- kratke iteracije, ki omogočajo hitro preverjanje in popravljanja,
- časovni okvir iteracije je od enega do šestih tednov,
- odstranitev vseh nepotrebnih aktivnosti,
- prilagodljivost na mogoča nepričakovana tveganja,
- inkrementalni pristop k procesu, ki omogoča izgradnjo rešitve v majhnih korakih,
- usklajen in inkrementalen pristop zmanjšuje tveganja,
- človeško naravnan agilni proces, ki daje prednost ljudem pred procesom in tehnologijo,
- sodelovalni in komunikativni delovni način.

Izbor ustreznih postopkov ni toliko usmerjen na omejevanje sprememb čim bolj zgodaj v projektu, ampak na boljše upravljanje s spremembami prek celotnega življenjskega cikla projekta.

Prednosti agilnega razvoja v primerjavi s klasičnimi metodologijami lahko povzamem kot:

- krajši razvoj sistema in krajši čas uvedbe,
- večja stabilnost in kakovost sistema zaradi zgodnjih povratnih informacij, pridobljenih od uporabnikov,
- boljša izkoriščenost ljudi in
- večja prilagodljivost spremembam (uporabniške zahteve in načrti vodstva).

Slabosti agilnega razvoja izhajajo iz zahtev, ki morajo biti izpolnjene pri uspešni gradnji. Te slabosti so naslednje:

- ne deluje pri velikem številu ljudi,
- ne deluje, ko je ekipa razdeljena na več oddaljenih lokacijah ali ko ni mogoča dobra komunikacija,
- zahteva kakovostne in izkušene člane ekip,
- zahteva visoko stopnjo vključenosti naročnika, kar ni vedno mogoče,
- zahteva visoke stroške testiranje, kljub temu da to bistveno poveča kakovost rešitve,
- na začetku ne obstaja jasna slika končnega izdelka.

V današnjem času so metode agilnega razvoja informacijskih sistemov med najbolj priljubljenimi. Ena izmed raziskav kaže, da večina organizacij še vedno ne uporablja nobene izmed formalnih metodologij, takoj za njimi pa se nahajajo organizacije, ki prisegajo na uporabo metod agilnega razvoja (West, 2010, str. 3). Prav te organizacije

imajo sposobnost povečanja inovativnosti in hitrosti pri ustvarjanju sprememb pred konkurenti. V svetu nenehnih sprememb je to ključnega pomena za uspeh.

Obstaja več modelov razvoja na področju agilnih metodologij. Najbolj znani predstavniki so model XP, model SCRUM, model FDD ter metoda DSDM. Zaradi uporabe metodologije DSDM v primeru razvoja, ki ga opisuje pričujoče delo, v nadaljevanju bolj podrobno podajam njen opis.

3.5 Izbor ustrezne metodologije

Problematika izbora ustrezne metodologije razvoja je bila v preteklosti velikokrat obravnavana tema. Številni avtorji podajajo ugotovitev, da ne obstaja univerzalna metodologija, ki bi bila učinkovita pri vseh projektih. Različni projekti imajo različne potrebe in značilnosti, ki se jih mora ob izboru metodologije upoštevati na način, da je tudi projekt učinkovit in uspešen.

Cockburn (2001) podaja odločitveni model, ki pomaga izbrati primerno metodologijo iz družine metodologij z imenom Crystal. Kljub temu da je model omejen na družino Crystal, pa se lahko uporablja tudi pri izbiri drugih metodologij. Odločitev za model temelji na treh glavnih lastnosti projekta:

- številu ljudi, vključenih v projekt,
- kritičnosti projekta in
- prednostnih nalogah projekta.

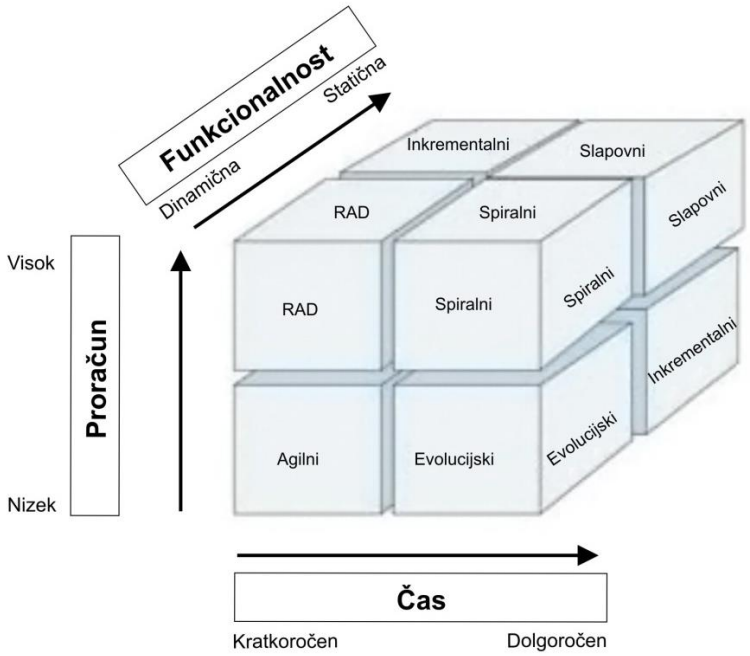
McConnell (1996, str. 154–157) predlaga bolj splošna navodila za izbor ustrezne metodologije, ki temelji na odgovorih na naslednja vprašanja:

- Kako dobro jaz in moje stranke razumemo zahteve na začetku projekta? Ali se bo naše razumevanje velikokrat spremenilo v času projekta?
- Kako dobro razumem sistemsko arhitekturo? Ali bodo potrebne velike arhitekturne spremembe med projektom?
- Kako zanesljiv sistem potrebujem?
- Koliko vnaprej moram načrtovati za prihodnje različice?
- Koliko tveganja vključuje projekt?
- Ali sem omejen po vnaprej določenem urniku?
- Ali moram imeti možnost za popravke znotraj posameznih faz projekta?
- Ali moram strankam zagotoviti pregled napredovanja projekta?
- Ali moram zagotoviti upravljanje napredovanja projekta?
- Koliko spretnosti je potrebno za uspešno uporabo izbrane metodologije?

Podjetje PM Solutions je izdelalo grafično ogrodje D^3 kocka, ki omogoča izbiro ustrezne metodologije. Vsebuje tri glavne kriterije, ki vplivajo na posamezen element poslovne vrednosti in v ustrezni kombinaciji predstavljajo osnovo izbire metodologije pri optimalni poslovni vrednosti. Kot prikazuje slika 1, so kriteriji:

- funkcionalnost, ki se lahko v času projekta zelo spreminja (dinamična) ali je natančno določena že na začetku projekta (statična),
- proračun, ki je odvisen od velikosti podjetja, povprečne velikosti projekta ter zahtev vrnitve investicije,
- čas, kjer se kratkoročna ali dolgoročna opredelitev naredi v skladu s tem, kaj poslovanje podjetja potrebuje za uspeh.

Slika 1: D^3 kocka



Vir: PM Solutions, Selecting a Software Development Life Cycle (SDLC), 2005.

Mnkandla (2008, str. 78–107) v svojem delu izdela ogrodje za izbor ustrezne metodologije s poudarkom na izboru agilnih metodologij. V njem upošteva številne ugotovitve in navodila za izbor metodologij ostalih avtorjev. Ogrodje sestavljajo tri stopnje. Na prvi se preveri, ali je projekt primeren za razvoj z uporabo pristopov agilnih metodologij. Pri tem se upošteva naslednje projektne parametre: stabilnost zahtev, velikost projekta, časovni okvir, kompleksnost projekta in tveganje projekta. Če ugotovimo, da je projekt primeren, se nadaljuje z drugo stopnjo, kjer se glede na metodološke značilnosti izbere ustrezne pristope izbrane skupine metodologij. Na tretji stopnji se izbrane pristope vpelje v projektno okolje.

Analiza literature in moje izkušnje kažejo, da imajo organizacije in IT-oddelki, ki se ukvarjajo z razvojem informacijskih sistemov, pogosto premalo poglobljenega znanja s področja metodologij razvoja. To ovira njihov izbor ustrezne metodologije in vodi v izbiro metodologije, ki jo poznajo ali je trenutno v uporabi. Izbrana metodologija tako vse prevečkrat ni optimalna za uporabo v trenutnem projektu, kar pa za organizacijo pomeni dodatne stroške ali v najslabšem primeru vodi celo v neuspeh projekta.

3.6 Metoda dinamičnega razvoja sistemov – model DSDM

Metoda dinamičnega razvoja sistemov (angl. *Dynamic System Development Method*) je model, ki ga je razvil konzorcij v Veliki Britaniji. Model je bil prvič objavljen leta 1994. Izvira iz RAD-metodologije hitrega razvoja aplikacij. Velikokrat je opredeljen kot prvi celovit model agilnega razvoja. Temeljna zamisel modela je, da v okviru razvoja določimo čas in vire, nato pa ustrezno prilagodimo še zahteve glede funkcionalnosti razvojnega produkta.

Prednosti modela se kažejo v naslednjih načelih, ki se jih model drži (DSDM Public Version 4.2 Manual, 2008):

- **Sodelovanje uporabnika je ključnega pomena.** Osebe, ki bodo uporabljale rešitev, morajo biti aktivno vključene v razvoj rešitve. To je ključno, da je lahko rešitev res uporabna za uporabnika, ki jo bo uporabljal.
- **Razvojna skupina mora imeti dovolj podpore, da lahko odloča.** Skupina mora imeti možnost sprejetja hitrih odločitev brez čakanja na njihovo odobritev prek vse prevečkrat toge birokracije.
- **Pogoste objave.** DSDM je osredotočen na pogoste objave rešitve. To omogoča prisotnost uporabnika v ključnih fazah razvoja. Prav tako omogoča hitro dostavo izdelka.
- **Izdelek mora ustrezati poslovnemu namenu.** Zagotavljanje poslovnemu namenu je pomembnejše od tehnične popolnosti.
- **Razvoj mora biti iterativen in inkrementalen.** Razvoj rešitve poteka v iteracijah, kar omogoča upoštevanje povratnih informacij uporabnikov. Rešitev se mora zagotoviti po delih, s čimer lahko uporabnik takoj prične z uporabo funkcionalnosti njenega dela. Več funkcionalnosti se doda v kasnejših ponovitvah.
- **Vse spremembe v razvoju so reverzibilne**. Vsi deli rešitve morajo biti ves čas v znanem stanju. To omogoča vzpostavitev prejšnjega stanja v primeru, da določene nove spremembe ne delujejo pravilno.
- **Zahteve so podane na visoki ravni.** Zahteve na visoki ravni so podane na začetku projekta. Bolj natančne podrobnosti se določi med razvojem dela funkcionalnosti.

- **Testira se znotraj življenjskega cikla**. Testiranje se izvaja na vsakem koraku. S tem se zagotovi, da je rešitev tehnično zanesljiva brez pomanjkljivosti in da se najboljšo uporabo doseže iz povratnih informacij uporabnikov.
- **Pomembno je sodelovanje vseh kakorkoli zainteresiranih znotraj projekta.** Sodelovanje med vsemi zainteresiranimi stranmi je bistvenega pomena za uspeh projekta. Vse vpletene strani (ne samo jedro ekipe) si morajo skupaj prizadevati za doseganje poslovnih ciljev.
- **Paretovo pravilo.** DSDM predpostavlja, da je mogoče 80 % rešitve razviti v 20 % časa, ki bi bil potreben za razvoj celotne rešitve. DSDM se osredotoča na teh 80 %, pri čemer 20 % pusti za razvoj v kasnejših revizijah. DSDM predpostavlja, da vse zahteve za končno rešitev niso znane, tako je zelo verjetno, da bi bilo 20 % nepomembnih funkcionalnosti kljub razvoju zahtev v celoti vseeno pomanjkljivo razvitih.

Življenjski cikel modela DSDM skupaj s predprojektnim ter poprojektnim delom vsebuje sedem faz (DSDM Public Version 4.2 Manual, 2008):

- **Predprojektna faza** se prične še pred uradnim začetkom projekta. V njej se opredeli poslovni problem, ki ga bomo reševali, določi okvir raziskovanja, vire ter sredstva, potrebna v fazi študije izvedljivosti, ter njen začetni načrt, umesti projekt v ustrezno strategijo ter vzpostavi začetno vodenje projekta.
- **Študija izvedljivosti,** v kateri se sprejme odločitev, ali uporabiti model DSDM za razvoj izdelka ali ne. Tu je treba odgovoriti na vprašanje, ali lahko projekt izvedemo znotraj omejitev časa in virov. To se določi s pomočjo presojanja tipa projekta, organizacije v podjetju in med ljudmi. Na koncu faze kot izid dobimo poročilo o izvedljivosti, začetni nepodrobni načrt razvoja in opredelitev glavnih tveganj.
- **Študija poslovanja** opredeli okvir poslovnega področja, ki ga bo rešitev zajemala. Na osnovi tega se določi lista zahtev po pomembnosti, načrt prototipa in razvoja, arhitekturo sistema, uporabljeno tehnologijo in dodatno opredeli tveganja.
- **Iteracija funkcionalnega modela** vključuje analizo, kodiranje in vključitev zahtevane funkcionalnosti v prototip. Ključen izid te faze je funkcionalen model, ki je sestavljen iz prototipa in analize modelov funkcionalnosti.
- **Iteracija načrtovanja in izgradnje** izpopolni funkcionalni prototip, da vsebuje tudi vse nefunkcionalne zahteve ter tako zadovolji zahteve uporabnikov. V tej fazi rešitev zgradimo s pomočjo načrtovanja, funkcionalnih prototipov ter na pripombah ter predlogih uporabnikov.
- **Implementacija** je faza, kjer se končna testirana rešitev iteracije izroči uporabnikom. Ob koncu faze se projekt vrne v naslednjo iteracijo funkcionalnega modela. Če smo razvili vse funkcionalne modele fazi sledi poprojektna faza.
- **P-projektna faza** vsebuje aktivnosti podpore in vzdrževanja rešitve. Temeljna naloga, ki jo mora faza zagotoviti, je, da je rešitev delujoča celotno obdobje njene uporabe.

Model DSDM opredeljuje ključne vloge, ki morajo biti dodeljene članom ekipe, vključene v razvoj (DSDM Public Version 4.2 Manual, 2008):

- **izvršni sponzor** je zagovornik sistema, ki ima sposobnost zagotoviti sredstva in vire, potrebne za projekt,
- **ambasador** je oseba, ki je ključna pri komunikaciji med uporabniki in razvojno ekipo. Koordinira razvojno ekipo in mora imeti dobro poznavanje delovanja celotnega sistema,
- **vizionar** je gonilna sila projekta. Ima največ znanja in pregleda nad projektom. Projekt usmerja in nadzoruje ter tako zagotavlja, da se vse dogaja v skladu z načrtom po poti do poslovnih ciljev,
- **svetovalec** je oseba, ki ima praktična znanja s področja poslovanja, ki ga rešitev vključuje,
- **razvijalec** je oseba, ki ima znanje in izkušnje, kako načrte in zahteve pretvoriti v delujočo kodo,
- **tehnični koordinator** usklajuje različne tehnične vidike sistema in zagotavlja njihovo pravilno medsebojno sodelovanje,
- **projektni vodja** nadzoruje razvoj in izdelavo prototipov,
- **koordinator** – olajšuje ekipno delo na praktičnih področjih,
- **pisatelj** je oseba, zadolžena za odločitve glede dokumentacije, razprav in načrtov ekipe,
- **vodja ekipe** skrbi za človeške vidike ekipe in zagotavlja, da ekipa deluje kot celota.

Obstaja več pristopov, ki jih DSDM priporoča pri razvoju rešitve. Tako proces načrtovanja vsebuje pristop izdelave prototipov, testiranja in modeliranja. Prav tako DSDM pri zajemanju zahtev priporoča pristop delavnic. Najpomembnejša pristopa z vidika zagotavljanja modela DSDM pa sta gotovo pristop pravila MoSCOW ter pristop časovnih okvirov. Ker so bili vsi omenjeni pristopi v veliki meri opisani v predhodnih poglavjih metodologij (predvsem poglavje opisa modela RAD, iz katerega model DSDM tudi izhaja), v nadaljevanju dodatno podajam le opis pristopa časovnih okvirov.

Pristop časovnih oken je izjemno pomemben del modela DSDM. Cilj pristopa je, da omogoči realizacijo rešitve v okviru fiksnega časa in stroškov. Pristop časovnih okvirov projekt v največji možni meri razdeli na manjše dele, pri čemer ima vsak del določen rok in stroške izvedbe. Tako edina preostala spremenljivka enačbe ostane »zahteve rešitve«, kar projektnemu vodji daje možnost opustitve najmanj pomembnih zahtev po pristopu MoSCOW. Na ta način DSDM prek načela Paretovega pravila zagotavlja dostavo rešitve v okviru predvidenega časa, denarja in funkcionalnosti (DSDM Public Version 4.2 Manual, 2008).

4 INFORMACIJSKI SISTEM FINANČNE INSTITUCIJE

4.1 Finančna institucija – lizing podjetje

Podjetje, za katerega se je razvila rešitev za upravljanje z likvidnostnim tveganjem, spada med kapitalske družbe. Registrirano je za osnovno dejavnost finančnega zakupa – lizing (angl. *leasing*). Lizing podjetje je finančna institucija, ki se ukvarja s finančnim posredništvom. Za finančno posredništvo je značilno prelivanje sredstev od strank s presežki na stran strank s primanjkljaji. Pri tem se finančno podjetje s pridobivanjem finančnih sredstev in prevzemanjem obveznosti za svoj račun izpostavlja tveganju.

Lizing podjetje lahko v splošnem ponuja dva tipa financiranja:

- **finančni lizing** je oblika financiranja, kjer se lizingojemalec že na začetku, ob sklepanju pogodbe, odloči za lastništvo nad predmetom lizinga po poteku pogodbe. Je najpogostejša oblika lizinga. Pomeni dejansko financiranje nakupa predmeta in je po svoji vsebini zelo podoben nakupu na obroke. Glavna značilnost je, da je predmet lizinga do končnega poplačila last lizingodajalca. Obračun obresti je navadno vezan na eno izmed referenčnih obrestnih mer, kar pomeni, da je višina prejetih obresti odvisna od tržnih vrednosti referenčnih obrestnih mer. Pogodbe ni mogoče odpovedati, lizingojemalec mora ob predčasni prekinitvi plačati sedanjo vrednost še neplačanih obrokov ter vse stroške vzdrževanja in zavarovanja predmeta,
- **operativni lizing** je posebna vrsta financiranja, namenjena vsem, ki jih zanima predvsem uporaba in ne toliko nakup predmeta financiranja po izteku pogodbenega razmerja. Lizingodajalec prejema od dajanja predmeta lizinga v najem tako imenovano najemnino v obliki enakih plačil mesečnih obrokov. Pogodbo je mogoče odpovedati brez večjih posledic za lizingojemalca. Vsa tveganja, povezana z lastništvom, kot tudi kritje vseh stroškov popravil, vzdrževanja in zavarovanja predmeta pogodbe nosi lizingodajalec.

Lizing podjetja so največkrat ustanovljena kot hčerinske družbe domačih ali tujih poslovnih bank. Za svojo dejavnost potrebujejo veliko finančnih sredstev. Ker niso banke, ne morejo sprejemati depozitov, zato se morajo financirati podobno kot vsako drugo podjetje. V praksi največkrat obstajajo lizing podjetja, ki na lizinškem trgu nastopajo prek lastnika, ki ima ugled na mednarodnih finančnih trgih. Razlog za to se nahaja predvsem v dejstvu, da si posojilodajalci želijo, da bi bili v primeru stečaja poplačani s kapitalom lastnika. Tako si lizing podjetje lažje zagotovi po obsegu večje in cenovno ugodnejše finančne vire, kar pa zanj pomeni tudi večji tržni delež in potencialno večji dobiček. Največje lizing hiše se zato v glavnem zadolžujejo prek bančnih trgov s pomočjo zastave dobrega imena matične banke, vendar tu nimajo prostih rok, saj jih po obsegu danih posojil omejuje Zakon o bančništvu. Zaradi tega so lizing podjetja primorana iskati vire financiranja tudi zunaj svoje bančne skupine.

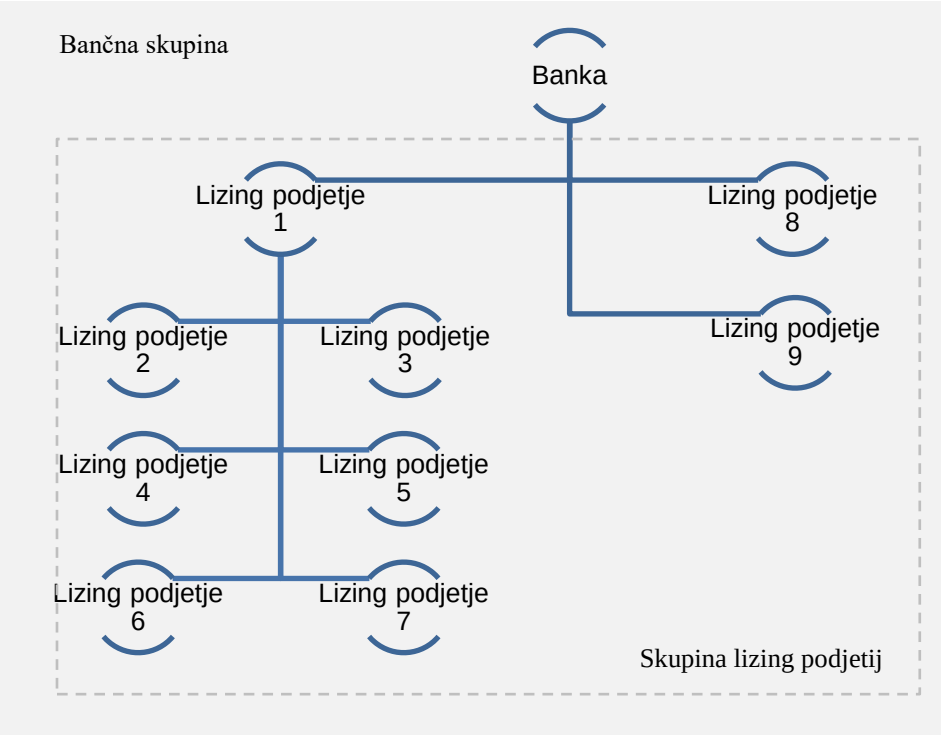
4.2 Likvidnostno tveganje lizing podjetja

Kot sem že omenil v prvem poglavju magistrskega dela, morajo finančne institucije za svoje nemoteno poslovanje imeti dovolj sredstev, saj v primeru njihovega pomanjkanja zelo hitro izgubijo svoje stranke in s tem tržni delež. Prav zato je treba paziti na pravilno razmerje med višino obveznosti do virov sredstev in višino investicij. Najpomembnejša z vidika likvidnostnega tveganja finančne institucije kot lizing podjetja je usklajenost med ročnostjo pogodb o lizingu in ročnostjo do virov sredstev. Lizingodajalec svoje obveznosti navadno poravna v veliko krajšem obdobju, kot jih poravna lizingojemalec. Prevelik razkorak lahko lizing podjetju zato povzroči hude likvidnostne probleme. Zaradi že omenjenih povezav lizing podjetja in matičnih bank kot njegovih glavnih financerjev je likvidnostno tveganje v veliki meri odvisno od sposobnosti bank za zagotavljanje virov financiranja svojim hčerinskim lizing podjetjem.

4.3 Lastniška in organizacijska struktura lizing podjetja

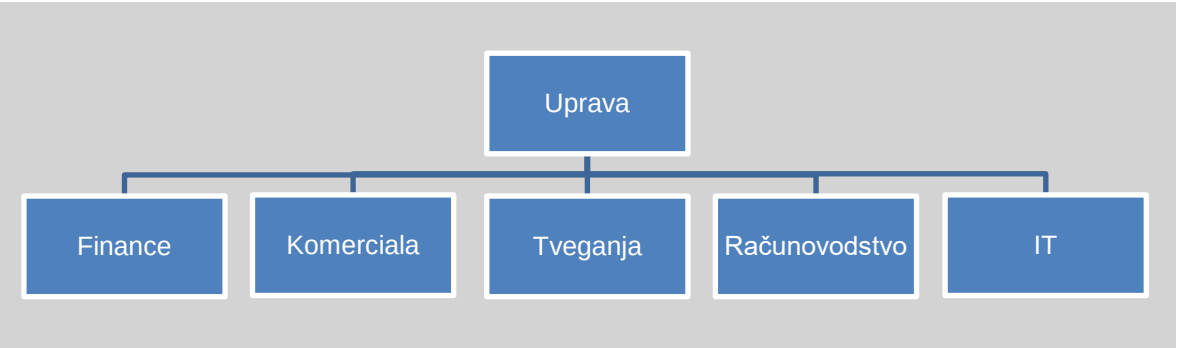
Podjetje z vplačilom osnovnega kapitala s strani edinega družbenika, ki se ukvarja z dejavnostjo bančništva, tvori bančno skupino. V okviru njenega delovanja občuti regulatorni vpliv, saj je matična banka močno podvržena regulatornim okvirom Banke Slovenije. Na področju različnih držav Balkana je podjetje ustanovilo šest hčerinskih podjetij. Skupaj še z dvema sestrskima lizing podjetjema predstavlja skupino lizing podjetij znotraj omenjene bančne skupine. Slika 2 prikazuje lastniške povezave znotraj bančne skupine. Zaradi zakrite identitete bančne skupine in lizing podjetij ta poimenujem s številkami od ena do devet. Podjetje, za katero se je rešitev razvila, je lizing podjetje 1. To podjetje je tudi nosilec razvoja lizing dejavnosti znotraj bančne skupine, kar je bilo opredeljeno s sprejetjem internih politik na področju razvoja lizing dejavnosti.

Slika 2: Lastniška struktura lizing podjetja



Organizacijsko strukturo podjetja prikazuje Slika 3. Ta govori o načinu povezanosti med nalogami in nosilci nalog znotraj podjetja. Izbrana mora biti optimalna oblika organizacijske strukture, ki vsebuje enote različnih velikosti in vsebin, tako da nam omogoča hitro in učinkovito komuniciranje med nosilci nalog.

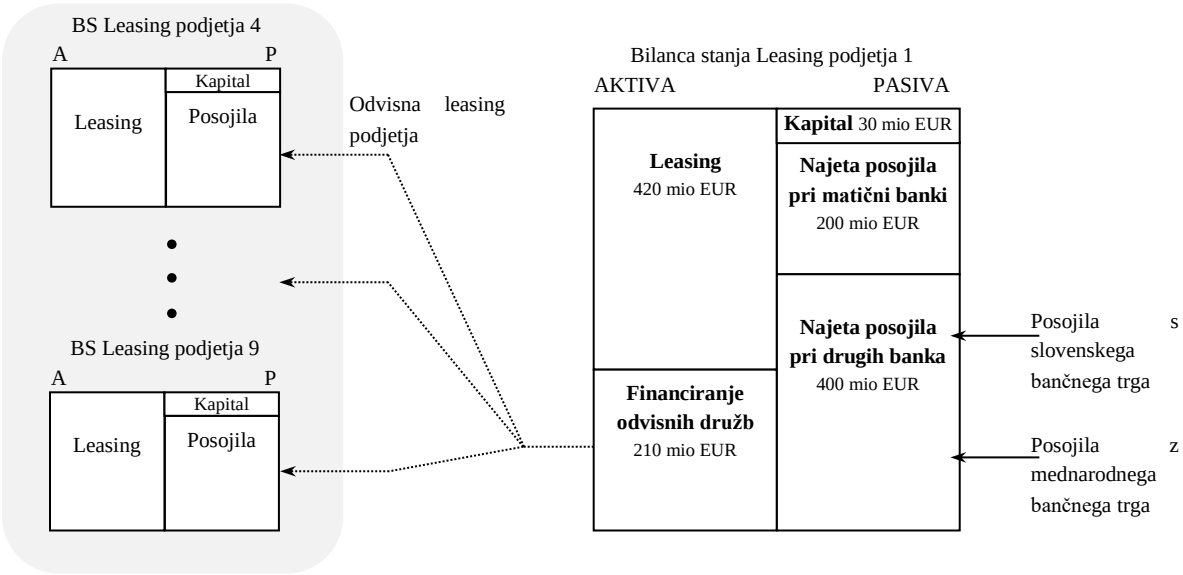
Slika 3: Organizacijska struktura lizing podjetja



4.4 Bilančna struktura lizing podjetja

Gradnja rešitve za spremljanje likvidnostnega tveganja izhaja iz bilančne strukture lizing podjetja. Slika 4 prikazuje bilančno vsoto kot tudi strukturo bilance stanja lizing podjetja. Zaradi lažje predstavitve upravljanja likvidnostnega tveganja je slika prilagojena in vsebuje le najpomembnejše predstavnike denarnih tokov.

Slika 4: Struktura bilance stanja lizing podjetja



Vir: S. Hajne, Razvoj informacijskega sistema za upravljanje z nekreditnimi tveganji, 2011, str. 20.

4.5 Trenutno stanje informacijskega sistema za podporo poslovanju

Vsa lizing podjetja znotraj bančne skupine za podporo poslovanju že uporabljajo obstoječ informacijski sistem Skupine podjetij Globus Marine International (v nadaljevanju GMI). Ta že od leta 1994 nudi najcelovitejšo podporo družbam, ki se ukvarjajo z dejavnostjo lizinga. Njihov informacijski sistem pokriva več kot 75 % trga dejavnosti lizinga na področju Slovenije, Hrvaške, Bosne in Hercegovine, Srbije, Črne Gore ter Makedonije (Kdo smo – Skupina podjetij Globus Marine International, 2011).

Njihov informacijski sistem funkcijsko pokriva celotno delovanje dejavnosti lizinga: od izdelave lizing ponudb do končne izdelave bilanc. Sestavljajo ga naslednji moduli:

- spremljava lizing poslov,
- računovodski paket z glavno knjigo,
- osnovna sredstva,
- posebne obdelave in
- avtomatska obdelava plačil.

Modul spremljave lizing poslov podpira vse znane in v praksi uporabljene načine financiranja. Pokriva naslednje funkcije v lizing podjetju:

- prodajo: kalkulacije, priprava ponudb, evidentiranje pogodb, spremljava pogodb, spremljava provizij, subvencij, spremljava izgradnje objektov, financiranje zalog,

- zaledno funkcijo: spremljava pogodb, različni načini reprogramiranja pogodb, fakturiranje, izterjava, opominjanje, spremljava plačil, izračun zamudnih obresti,
- računovodstvo: avtomatsko evidentiranje dogodkov v knjigovodstvu, davčne napovedi, avtomatska povezava z glavno knjigo,
- ocenjevanje tveganja: zajem ključnih podatkov, vrednotenje partnerjev, odobritve, okviri financiranja,
- kontroling: podpora poročanju in statistikam,
- finance: statistični pregledi.

Računovodski paket z glavno knjigo nudi vse standardne funkcionalnosti glavne knjige:

- spremljanje saldakontov,
- izdelavo bilanc,
- kontne načrte,
- fakturiranje in spremljavo vhodnih ter izhodnih računov,
- obračun DDV.

Modul »osnovna sredstva« pokriva naslednje funkcije:

- obračun poslovne amortizacije,
- obračun davčne amortizacije po različnih metodah,
- računovodsko analitiko vknjižb osnovnih sredstev,
- avtomatsko računovodsko evidentiranje in
- odpise ter prodajo osnovnih sredstev.

Modul posebnih obdelav je namenjen predvsem spremljanju izterjav in služi kot dopolnilni modul k modulu »spremljava lizing poslov«.

Modul »avtomatska obdelava plačil« je namenjen povezavi s sistemom elektronskega bančništva ter izmenjavi podatkov o prejetih in danih plačilih.

Iz opisa funkcionalnosti GMI informacijskega sistema je razvidno, da sistem zagotavlja le podporo tistemu delu poslovanja, ki je neposredno vezan na spremljavo lizing poslov. Kot je razvidno s Slike 4, ta predstavlja dve tretjini (420 mio. EUR) celotne bilančne vsote podjetja (630 mio. EUR). Pomanjkljivo je predvsem spremljanje danih posojil odvisnim družbam (230 mio. EUR) in prejetih posojil od bank (630 mio. EUR). Uspešno upravljanje z likvidnostnim tveganjem pa zahteva razpolago z natančnimi podatki iz amortizacijskih načrtov poslov tako na aktivni kot tudi na pasivni strani bilance stanja. V nasprotnem primeru pride do neučinkovitega upravljanja likvidnostnega tveganja, kar privede do nepotrebnih stroškov in stroškovno neoptimalnih delovnih procesov.

4.6 Ciljno stanje informacijskega sistema za podporo poslovanju

Imeti je treba jasno vizijo, kaj je ciljno stanje in kakšen informacijski sistem bomo imeli. V začetnem poglavju sem podal pomembnost prisotnosti sistema za upravljanje z likvidnostnim tveganjem v podjetju pri doseganju uspešnosti in učinkovitosti. V omenjenem lizing podjetju informacijska rešitev, ki bi nudila ustrezno podporo upravljanju z omenjenim likvidnostnim tveganjem, ni obstajala. Treba jo je bilo zagotoviti.

Sprva se je v podjetju porodila ideja, da bi se vzpostavilo le ustrezno podatkovno skladišče, ki bi obstoječe podatke poslovnega sistema smiselno uredilo in strukturiralo ter tako nudilo učinkovito ter ažurno poizvedbo za namen managementa likvidnostnega tveganja. Že v opisu trenutnega stanja pa sem podal ugotovitev, da moramo za učinkovito in uspešno upravljanje likvidnostnega tveganja razpolagati z natančnimi podatki amortizacijskih načrtov poslov tako na aktivni kot na pasivni strani bilance. Trenutni poslovni sistem pa ni vseboval spremljanja danih posojil odvisnim družbam ter prejetih posojil od bank. Tako so odgovorni v podjetju spoznali, da zagotovitev podatkovnega skladišča ne bo dovolj ter bo treba pridobiti novo rešitev, ki bi omogočala vnos vseh manjkajočih poslovnih podatkov. Skupaj z obstoječim poslovnim sistemom bi nato zagotovila učinkovit pregled in upravljanje likvidnostnega tveganja.

5 RAZVOJ INFORMACIJSKEGA SISTEMA ZA UPRAVLJANJE LIKVIDNOSTNEGA TVEGANJA V LIZING PODJETJU

V lizing podjetju so se zaradi spoznanja, da nobena rešitev na trgu ni v celoti zadovoljila specifičnih zahtev podjetja, odločili za zasnovo in razvoj nove informacijske rešitve. Na Sliki 2 iz prejšnjega poglavja smo videli, da podjetje vsebuje organizacijsko enoto za IT. Povzete naloge omenjene enote, ki so natančno opredeljene v internih dokumentih lizing podjetja, so:

- načrtovanje in izvajanje informatizacije v podjetju,
- priprava, vzdrževanje in izvajanje strategije IT,
- zagotavljanje tekočega delovanja informacijske infrastrukture,
- načrtovanje in izvajanje postopkov za zagotavljanje varnosti informacijskega sistema,
- vodenje projektov s področja IT in
- pripravljanje in uvajanje standardov IT.

S samostojnim razvojem novih informacijskih sistemov se enota ne ukvarja. Ekipa znotraj enote je številčno zelo majhna. Vsakodnevno opravljanje obstoječih nalog ji vzame preveč časa, da bi se lotila tudi razvoja novega sistema. Zaposleni znotraj enote tudi nimajo potrebnih znanj in izkušenj, ki so pri razvoju novega informacijskega sistema še kako potrebni. To so bili glavni razlogi, da se je v podjetju sprejela odločitev za zunanje

izvajanje razvoja informacijske rešitve. Prednosti in slabosti zunanjega izvajanja sem opisal v poglavju 2.4.

Kot zunanjega izvajalca so v lizing podjetju izbrali podjetje, katerega primarna dejavnost je bila ukvarjanje z IT. Dva izmed ključnih dejavnikov izbora ustreznega IT-podjetja sta bila že uspešno sodelovanje in zaupanje med podjetjema v preteklih projektih. Nabor storitev IT-podjetja je vseboval razvoj informacijskih sistemov na ključ ter na ta način lizing podjetju zagotavljal specializacijo ter izkušnje z omenjenega področja razvoja. Kot zaposleni v IT-podjetju, ki je zagotavljalo zunanje izvajanje razvoja, ter kot udeleženec v razvoju sem nastopal tudi sam.

5.1 Izbor metodologije – pred-projektna faza

Izbor metodologije razvoja sistema v lizing podjetju je zaradi zunanjega izvajanja razvoja temeljil na znanjih in izkušnjah zunanjega izvajalca – IT-podjetja, ob upoštevanju posameznih značilnosti lizing podjetja kot naročnika razvoja ter v dogovoru z njim. Za prve smernice izbora ustrezne metodologije se je uporabilo ogrodje D^3 kocke, ki je opisano v poglavju 3.5. Ogrodje zahteva določitev kriterijev proračuna, časa in funkcionalnosti.

Proračun projekta se je ocenil glede na proračun obstoječih IT-projektov lizing podjetja. Postavilo se je območje, ki se je začelo z najcenejšim projektom (10.000 EUR) ter končalo z najdražjim projektom (400.000 EUR) podjetja. Glede na večje število projektov z nižjimi zneski se za mejno točko, ki določa vrsto projekta glede na proračun, ni vzela sredina (200.000 EUR), ampak vrednost 150.000 EUR. To je za lizing podjetje predstavljalo mejni znesek med nizkim in visokim tveganjem podjetja. Za projekt upravljanja likvidnosti lizing podjetja je bilo že ob njegovi vzpostavitvi opredeljenih 100.000 EUR sredstev. Glede na mejno točko se je projekt opredelil kot nizkoproračunski.

Čas projekta se je opredelil glede na časovno mejo, ki se je izračunala po enačbi:

$$TH = (SD_L - LD_S)/4 + LD_S,$$

pri čemer je

- TH – časovna meja,
- SD_L – najkrajše trajanje najdaljšega projekta (če pričakujemo, da bo trajanje projekta 900 dni, vendar se optimistično lahko zaključi v 700 dneh, potem je $SD_L = 700$),
- LD_S – najdaljše trajanje najkrajšega projekta (če pričakujemo, da se naš najkrajši projekt zaključi v 30 dneh, vendar se pesimistično lahko zaključi v 90 dneh, potem je $LD_S = 90$),
- v primeru lizing podjetja je bila tako izračunana časovna meja:

$$TH = (700 - 90)/4 + 90 = 243 \text{ dni}$$

Projekt razvoja sistema managementa likvidnostnega tveganja je bil časovno omejen na 180 dni in tako glede na izračunano časovno mejo opredeljen kot kratkoročni projekt.

Funkcionalnosti sistema za management likvidnostnega tveganja ni bilo mogoče do potankosti določiti že na začetku projekta. V času projekta je potekal tudi vzporedni projekt, ki je vplival na zahteve in funkcionalnosti tega projekta. Tako smo s stališča funkcionalnosti, v ogrodju D^3 kocke, to opredelili kot dinamično.

Pridobljene ocene posameznih kriterijev D^3 kocke za opazovani projekt so pokazale, da je za razvoj sistema najprimernejša uporaba agilne metodologije. Ker je skupina agilnih metodologij velika, se je bilo treba odločiti, katera od metodologij znotraj skupine je najprimernejša. V IT-podjetju, ki je opravljalo storitve razvoja, smo že imeli dobre izkušnje z uporabo pristopov Scrum, XP in DSDM. Pri nadaljnjem odločanju o ustrezni metodologiji smo se zato omejili na izbor ene izmed njih.

V dogovoru z lizing podjetjem smo se odločili za uporabo pristopov metodologije DSDM. Odločitev za to se je sprejela na podlagi naslednjih kriterijev:

- **časovna omejitev projekta**: za IS je bilo že na začetku razvoja določeno obdobje, ko mora sistem biti na voljo za uporabo. Ena izmed funkcionalnih zahtev sistema so bila različna poročila, ki jih je moralo lizing podjetje od določenega obdobja naprej posredovati matični banki, da bi ta zadostila regulativi. DSDM-metodologija je agilna metodologija, ki zahteva fiksno časovno obdobje razvoja. V okviru tega nato prilagaja funkcionalne zahteve rešitve,
- **inkrementalni razvoj**: predvidelo se je obdobje sedmih mesecev, v katerem bi bilo pravilno zasnovano in razvito ogrodje sistema skupaj z začetnimi funkcionalnostmi. Te so bile v veliki meri odvisne od trenutnih potreb lizing podjetja. Razvoj vseh ostalih funkcionalnosti bo potekal glede nadaljnje potrebe in prioritete lizing podjetja v časovnih okvirih dveh mesecev,
- **dokumentacija:** lizing podjetje je kot finančna institucija izpostavljeno različnim regulativam glede dokumentiranosti sistema in kasnejših revizij. Ker so agilne metodologije zelo osredotočene na rešitev, je njihova skupna značilnost zelo pomanjkljiva dokumentiranost novega izdelka. DSDM-metodologija pa je metodologija agilnega razvoja, ki predvideva tudi izdelavo različnih tipov dokumentov prek posameznih faz metodologije,
- **omejitev človeških virov projekta**: XP-metodologija zahteva uporabo principa programiranja v paru. Zaradi že prevelike zasedenosti urnikov človeških virov, ki so bili potencialni udeleženci na tem projektu, te zahteve ne bi bilo mogoče zagotoviti.

Po posvetu strokovnjakov lizing podjetja in IT-podjetja, ki je opravljalo storitve razvoja, se je menilo, da bo projekt najbolje potekal z uporabo naslednjih devetih načel, ki jih predpostavlja DSDM-metodologija in so podrobneje opisana v poglavju 3.6:

- sodelovanje uporabnika,
- razvojna skupina mora imeti dovolj podpore, da lahko odloča,
- pogoste objave,
- izdelek mora ustrezati poslovnemu namenu,
- razvoj mora biti iterativen in inkrementalen,
- vse spremembe v razvoju so reverzibilne,
- zahteve so podane na visoki ravni,
- testira se znotraj življenjskega cikla,
- pomembno je sodelovanje vseh kakorkoli zainteresiranih znotraj projekta.

Nadalje se je odločilo za naslednje tehnike in pristope DSDM-metodologije razvoja informacijskih sistemov, ki so podrobneje opisani v poglavjih 3.3–3.6:

- pristop časovnih okvirov, ki projekt razdeli na več časovnih delov,
- MoSCoW-pristop, ki postavi zahteve v prednostno vrsto,
- prototipiranje, ki omogoča zgodnje odkrivanje pomanjkljivosti sistema in spremljanje odzivov prihodnjih uporabnikov,
- testiranje prek iteracij, ki zagotavlja visoko raven kakovosti,
- sestankovanje, kjer se vsi udeleženci projekta pogovorijo o zahtevah in funkcionalnosti sistema,
- projektno vodenje, ki spremlja potek projekta od začetka do njegovega zaključka.

Glede na zahteve DSDM-metodologije, ki sem jih podrobneje opisal v poglavju 5.6, so se dodelile ključne vloge udeležencem v projektu. Zaradi omejenih virov, namenjenih projektu razvoja, posameznih vlog DSDM-metodologije niso zastopale različne osebe. V razvoj informacijskega sistema so bile tako vključene štiri osebe z naslednjimi vlogami:

1. direktor lizing podjetja: izvršni sponzor,
2. vodja informatike lizing podjetja: ambasador, projektni vodja, koordinator, vodja ekipe,
3. poslovni analitik: vizionar, svetovalec, pisatelj,
4. zunanji izvajalec: razvijalec, tehnični koordinator, pisatelj.

V nadaljevanju poglavja podajam opis ključnih faz življenjskega cikla DSDM-modela razvoja.

5.2 Študija izvedljivosti

V tej fazi se je preverilo primernost različnih razvojnih tehnik in življenjskega cikla DSDM ter odgovorilo na vprašanje, ali lahko projekt izvedemo znotraj omejitev časa in virov. To se je določilo s pomočjo presojanja tipa projekta, organizacije v podjetju in med ljudmi. Na koncu faze smo kot izid dobili poročilo o izvedljivosti, začetni nepodrobni načrt razvoja in opredelitev glavnih tveganj.

5.2.1 Sposobnost projekta, da zadovolji poslovne potrebe

Podjetje je po raziskavi trga ugotovilo, da ne obstaja ustrezna rešitev, ki bi v celoti zadovoljila specifične potrebe lizing podjetja. Ovire za nakup že izdelanega sistema so bile predvsem:

- pomanjkljiva funkcionalnost že izdelanega sistema oziroma specifične potrebe podjetja,
- visoka cena in
- pogoji njegovega vzdrževanja.

Kot sem omenil že v poglavju 4.5 o oceni trenutnega stanja informacijskega sistema, je treba za učinkovito in uspešno upravljanje likvidnostnega tveganja razpolagati z natančnimi podatki amortizacijskih načrtov poslov tako na aktivni kot na pasivni strani bilance. Trenutni poslovni sistem pa ni vseboval spremljanja danih posojil odvisnim družbam in prejetih posojil od bank.

Projekt mora tako zagotoviti naslednje tri ključne funkcionalnosti:

- sistem mora omogočati vnos vseh manjkajočih poslovnih podatkov, ki so potrebni za učinkovit pregled in management likvidnostnega tveganja,
- sistem mora skupaj z obstoječim poslovnim sistemom zagotoviti učinkovit pregled in management likvidnostnega tveganja,
- sistem mora biti dovolj prilagodljiv glede na povratne potrebe uporabnikov, predvsem glede novih tipov pregledov.

5.2.2 Primernost projekta za DSDM-model razvoja

Kot sem že opisal v poglavju 5.1, kjer sem govoril o izbiri ustrezne metodologije razvoja, so bili ključni kriteriji za izbor metodologije DSDM časovna omejitev projekta, inkrementalni razvoj, dokumentacija in omejitev človeških virov projekta. Ker mora biti končni produkt tudi dovolj prilagodljiv glede na nove povratne zahteve uporabnikov, razvoj sistema zahteva ustrezno modeliranje in upravljanje teh sprememb.

DSDM nam omogoča pogled na sistem kot celoto. Omogoča nam razčlenitev zahtev na način, da je vsako nadaljnje delo z lahkoto povezano v celoto. Stranka lahko z uporabo prototipiranja predhodno preizkusi funkcionalnosti in tako poda smernice za naslednje časovne okvirje. Dejstvo, da sistem predvideva prilagodljivost in spremembe, samo potrdi uporabo metodologije, kot je DSDM.

5.2.3 Tveganja

Glavno tveganje projekta je tveganje neuspešnega končanja izdelka. Nadalje smo ugotovili, da je tveganje vsak dogodek, ki povzroča kakršnokoli motnjo, ki onemogoča uspešno izdelavo v okviru načrtovanega časa.

Da bi znali pravilno identificirati vse vrste tveganj, ki lahko nastopijo, ter obenem najbolj učinkovito omejili njihove pojave, smo proces upravljanja s tveganji razdelili na naslednje tri faze:

- opredelitev tveganj, kjer smo zaznali vsa morebitna tveganja, do katerih lahko pride med projektom,
- ocena tveganj, kjer smo vsakemu opredeljenemu tveganju prejšnje faze podali ustrezno prioriteto vpliva na dosego cilja ter verjetnost pojavitve v projektu,
- načrt ravnanja s tveganji, kjer opredelimo aktivnosti za ublažitev tveganj.

5.2.3.1 Opredelitev tveganj

Prva faza upravljanja s tveganji je njihova opredelitev. Pri projektu razvoja sistema za management likvidnostnega tveganja smo identificirali naslednje vrste tveganj:

1. **obveznosti članov**: tveganje je povezano z obveznostnimi, ki jih imajo člani projektne skupine zunaj projekta, kot so drugo delo v podjetju, družina ipd. Primer: nenadno povečanje obsega del enega izmed članov na njegovem delovnem mestu,
2. **usposobljenost zunanjega izvajalca**: ker je del projekta opravljal zunanji izvajalec, se pojavlja tveganje, da zunanji izvajalec ni dovolj strokoven ali usposobljen,
3. **kompleksnost kodiranja**: tveganje je povezano z znanjem programskega jezika, algoritmov ali drugih vidikov, ki zahtevajo visoko raven razumevanja,
4. **integracija kode**: tveganje je povezano s problemi, s katerimi se srečujemo pri integraciji sistema zaradi različnih stilov kodiranja, standardov ipd.,
5. **okvara strojne opreme**: se nanaša na tveganje v povezavi z ustavitvijo in izgubo podatkov zaradi okvare razvojne strojne opreme. Primer: trdi disk se okvari, kar povzroči zamude v razvoju,
6. **zdravje**: tveganje v povezavi z zdravstvenim stanjem udeležencev v projektu, ki lahko vplivajo na učinkovitost, dostopnost in včasih tudi sposobnost za dokončanje projekta,

7. **konflikti**: tveganje, povezano s človeškimi medosebnimi odnosi. Vsak član je drugačen, edinstven in ima svoje predstave o reševanju težav. Vsi se ne poznajo med seboj, še posebej v povezavi z zunanjim izvajalcem,
8. **spreminjajoče se zahteve**: tveganje v povezavi z nenehnim spreminjanjem zahtev pri dosegu želenih rezultatov. Lahko nastane zaradi slabega razumevanja zahtev,
9. **nova razvojna orodja**: tveganje v povezavi z novimi izdajami razvojnih orodij, ki bi lahko bila nezdružljiva s sedanjimi.

5.2.3.2 Ocena tveganj

V fazi ocene tveganja smo vsakemu opredeljenemu tveganju dodelili prioriteto glede na to, kako bi tveganja vplivala na dosego cilja ter verjetnost pojava tveganja v projektu. Pri tem smo za mere prioritete od zgoraj navzdol uporabili katastrofalna, ključna in pomembna, za mere verjetnosti od zgoraj navzdol pa visoka, srednja in nizka tveganja.

Tabela 1: Ocena tveganja

Tveganje	Prioriteta	Verjetnost
Obveznosti članov	Pomembna	Srednja
Usposobljenost zunanjega izvajalca	Katastrofalna	Nizka
Kompleksnost kodiranja	Ključna	Srednja
Integracija kode	Ključna	Srednja
Okvara strojne opreme	Katastrofalna	Nizka
Zdravje	Pomembna	Srednja
Konflikti	Pomembna	Nizka
Spreminjajoče zahteve	Pomembna	Nizka
Nova razvojna orodja	Katastrofalna	Visoka

5.2.3.3 Načrt ravnanja s tveganji

Po ovrednotenju in oceni tveganj se je sprejelo načrt, kako ravnati s posameznimi tveganji. Pri tem se je uporabilo naslednja štiri načela:

- **preprečevanje** tveganja, tako da odpravimo aktivnosti, ki povzročajo tveganje.
- **zmanjševanje** tveganja, tako da uporabimo metode, ki zmanjšajo tveganje,
- **sprejetje** tveganja, kjer sprejmemo tveganje in tudi škodo, ki se pojavi (primerna za manjša tveganja),
- **prenos** tveganja, kjer tveganje prenesemo na kakšen drug del sistema v podjetju.

Tabela 2: Načrt ravnanja s tveganji

Tveganje	Načelo	Opis
Obveznosti članov	Zmanjšanje, prenos	Drugim obveznostim v življenju se nikoli ni mogoče povsem izogniti, zato moramo biti sposobni pametno upravljati čas. Pristop, ki se je uporabil, je iskanje rešitev z razumevanjem članov, vključenih v projekt. Poleg tega je nekatera dela opravljal zunanji izvajalec, ki je tako del tveganja prenesel nase. Skupaj z upravljanjem časa se zato lahko tveganje bistveno zmanjša.
Usposobljenost zunanjega izvajalca	Zmanjšanje	Z izborom znanega zunanjega izvajalca, s katerim je podjetje že uspešno sodelovalo v podobnih projektih, se je tveganje bistveno zmanjšalo.
Kompleksnost kodiranja	Zmanjšanje	Vsak projekt, ki vsebuje kodiranje, vsebuje to tveganje. S sprejetjem ustreznega programskega jezika, ki zahteva doslednost in strokovnega zunanjega izvajalca, ki zagotavlja najvišje standarde kakovosti kodiranja, se je to tveganje zelo zmanjšalo.
Integracija kode	Preprečevanje	Tveganje smo popolnoma odpravili z ustreznim načrtovanjem ter uporabo odprtih protokolov ter standardov še pred pričetkom kodiranja.
Okvara strojne opreme	Zmanjšanje	Nepredvidljive strojne napake lahko pomenijo izgubo mesecev dela. S sprejetjem pravil vsakodnevnega shranjevanja kode v shrambo kode, ki je imelo omogočeno varnostno kopiranje, smo to tveganje zelo zmanjšali.
Zdravje	Sprejetje, preprečevanje	Tveganju se lahko v veliki meri izognemo s pravilno prehrano in počitkom. To pa je zelo težavno, ker so člani tima vpeti med delom, družino in študijske obveznosti. Zato je bilo edino smiselno, da se tveganje sprejeme in delo v primeru bolezni dodeli drugim članom.

Se nadaljuje

nadaljevanje

Tveganje	Načelo	Opis
Konflikti	Sprejetje, prenos	Naša čustva so ena izmed najbolj zapletenih stvari. Stopnja tolerance se razlikuje od osebe do osebe. Treba se je bilo sprijazniti, da je nemogoče ustreči vsem, zato smo se člani projekta že prej dogovorili, da bomo odkrito izražali mnenje brez zadržkov ter tako vzpostavili zdravo klimo sodelovanja, ki je zelo pomembna za uspeh projekta. Poleg tega se je sklenilo, da se v primeru nesoglasij poišče nasvet projektnega vodje in skupaj poišče rešitev. Tveganje v povezavi s konflikti s člani zunanjega izvajalca pa se je bistveno zmanjšalo, ker je bil izbran že poznan zunanji izvajalec z znanimi člani, ki so že imeli reference dobrega medsebojnega sodelovanja.
Spreminjajoče zahteve	Sprejetje	Ne glede na še tako poglobljeno načrtovanje med projektom slej ko prej prihaja do spremembe zahtev. Zato smo to sprejeli in se obenem zavedali, da nam povratne informacije uporabnikov, ki povzročajo spremembe, včasih lahko bistveno izboljšajo sistem.
Nova razvojna orodja	Preprečevanje	Temu smo se popolnoma izognili z vztrajanjem pri eni različici razvojnega orodja.

5.2.4 Prototip izvedljivosti

Na koncu faze se opredelijo glavni kriteriji za uspešnost projekta. Te opredelimo le površinsko, saj nam služijo le kot smernice, kaj želimo doseči na koncu. Podrobnejši seznam kriterijev se izoblikuje v nadaljnjih fazah.

Za omenjeni projekt smo določili naslednje kriterije uspešnosti:

- vnos vseh manjkajočih poslovnih podatkov, ki so potrebni za učinkovit pregled in management likvidnostnega tveganja,
- učinkovit pregled in management likvidnostnega tveganja,

- robustna zasnova, ki omogoča preprosto širitev in spremembe obstoječega sistema,
- dobra uporabniška izkušnja, ki omogoča enostavno navigacijo in preprostost z najmanjšim naporom pri izvajanju nalog.

5.3 Študija poslovanja

V fazi študija poslovanja smo opredelili okvir poslovnega področja, ki ga bo rešitev zajemala. Na osnovi tega so se določili lista zahtev po pomembnosti, načrt prototipa ter razvoja, arhitektura sistema in uporabljena tehnologija.

5.3.1 Definicija poslovnega prostora

S procesom zbiranja podatkov je poslovni analitik, ki je bil hkrati tudi idejni vodja projekta, s pomočjo strokovne literature definiral vse denarne tokove, ki jih je treba upoštevati pri upravljanju likvidnostnega tveganja podjetja. Pri tem je v veliki meri upošteval tudi izkušnje, ki jih je že pridobil pri opravljanju operativnega dela in pri analizi težav, s katerimi se je soočalo podjetje na področju finančnega upravljanja.

Zaznal je naslednje poslovne gradnike, ki bi jih bilo treba izdelati v prvem inkrementu. Vsak predstavlja enega izmed denarnih tokov in tako vplivajo na upravljanje likvidnostnega tveganja:

- **amortizacijski načrti najetih posojil** – ker je lizing podjetje svojo lizing dejavnost financiralo prek zadolževanj na domačem in tujem bančnem trgu je za učinkovito spremljanje likvidnosti treba natančno poznati načrt predvidenih plačil podjetja, ki bo popolnoma usklajen z amortizacijskimi načrti banke. To pomeni, da bodo pričakovane zapadlosti obveznosti podjetja enake pričakovanim zapadlostim terjatev bank. Za lizing podjetje v fazi odplačevanja predstavljajo odlive denarnih sredstev,
- **najave komercialistov o predvidenih naložbah** – lizing podjetja prodajajo lizing produkte prek prodajne mreže komercialistov, ki se nahajajo v poslovalnicah. Komercialist na osnovi izkušenj in ustrezne analize predložene dokumentacije za potencialnega lizingojemalca oceni možnost financiranja. Če ga komercialist oceni za plačilno sposobnega, v imenu podjetja z njim sklene lizing pogodbo. Za zagotovitev potrebnih denarnih sredstev mora komercialist čim prej najaviti višino odliva sredstev. Do sedaj so se najave zbirale prek elektronske pošte. Za lizing podjetje najava predstavlja odliv denarnih sredstev,
- **najave odvisnih družb za črpanje posojil** – odvisne družbe so lizing dejavnost financirale prek zadolževanja pri matičnem lizing podjetju. Obvladujoče podjetje je odvisnim družbam odobrilo posojila z možnostjo sukcesivnega črpanja s predhodno najavo (3 dni). Do sedaj so se najave zbirale prek elektronske pošte in telefonskih klicev. Za lizing podjetje je najava predstavljala odliv sredstev,

- **operativni stroški** – podjetje med operativne stroške šteje stroške dela in stroške blaga ter storitev. Pri učinkovitem upravljanju likvidnostnega tveganja je treba upoštevati zapadlosti operativnih stroškov in zagotavljati potreben obseg denarnih sredstev za njihovo nemoteno plačilo. Za lizing podjetje predstavljajo odlive denarnih sredstev. Vodijo se v poslovnem sistemu lizing podjetja,
- **obrestni izvedeni finančni instrumenti** – so izvedeni finančni instrumenti, ki so odvisni od gibanja obrestnih mer. Lizing podjetje je z najetimi posojili, vezanimi na variabilno obrestno mero, financiralo operativni lizing, ki je vezan na fiksno obrestno mero. Z namenom varovanja je lizing podjetje od matične banke kupilo klasično zamenjavo obrestnih mer, s katero je preoblikovalo variabilna plačila obresti za najeta posojila v fiksna. Tako je podjetje fiksiralo stroške financiranja za ves čas trajanja kredita, četudi bi obrestne mere v prihodnosti naraščale. Neto denarni tok je odvisen od ravni pogodbeno določene fiksne obrestne mere in tržne vrednosti referenčne obrestne mere,
- **amortizacijski načrti plačil lizing terjatev** – se vodijo v poslovnem sistemu lizing podjetja. Poslovni sistem vsaki lizing terjatvi na podlagi pretekle poplačljivosti določi bonitetno oceno, ki signalizira časovni razkorak med pogodbeno zapadlostjo terjatve in dejanskim plačilom terjatve. Terjatve, ki se plačujejo do 15 dni od njihove pogodbene zapadlosti, predstavljajo tiste denarne prilive, ki jih je smiselno upoštevati pri dnevnem načrtovanju denarnih tokov. Ko podjetje v svoje predvidene denarne prilive vključuje tudi terjatve, ki so v preteklosti zamujale več kot 15 dni, lahko na letni ravni podceni potrebo po refinanciranju. Za lizing podjetje tip denarnega toka predstavlja priliv denarnih sredstev,
- **dodatna zadolževanja pri bankah** – pomenijo povečanje denarnih sredstev na poslovnem računu podjetja. Čas od odločitve podjetja za najem bančnega posojila do realizacije njegovega črpanja lahko zelo variira in je odvisen od banke in vrste posojila. Banke v povprečju potrebujejo okoli štiri tedne, preden podjetje pride do dodatnih denarnih sredstev, s katerimi financira nove naložbe. Finančniki morajo zato skrbno načrtovati prihodnje denarne tokove in glede svoje njihove projekcije pravočasno intenzivirati aktivnosti zadolževanja,
- **amortizacijski načrt revolving posojil** – revolving posojilo je kombinacija kratkoročnega posojila in limita na transakcijskem računu ter predstavlja rešitev za financiranje občasnih likvidnostnih težav podjetij. Tudi lizing podjetje se je odločalo za uporabo tega instrumenta. Njegova prednost, če ga primerjamo z navadnim posojilom, je v tem, da lahko podjetje z enodnevno predhodno najavo črpa ali vrača denarna sredstva v okviru pogodbeno dogovorjenega zneska,
- **amortizacijski načrt vezave denarnih sredstev** – podjetje veže presežna denarna sredstva pri banki v primeru, ko nima možnosti vračil v okviru najetih revolving posojil ali ko nima ustreznih možnosti za sklenitev novih lizing naložb,
- **drugi denarni tokovi** – ki materialno vplivajo na likvidnostni položaj podjetja. Lizing podjetje ima odvisne družbe, ki izplačujejo dividende, njihova višina pa vpliva na

denarni tok. Drugi primer so dokapitalizacije odvisnih družb, ki jih načrtujemo prek omenjenega atributa denarnega toka.

Po proučitvi glavnih poslovnih gradnikov, postavljenih s strani poslovnega analitika, smo skupaj določili naslednje dodatne zahteve, ki jih mora informacijska rešitev vsebovati:

- informacijska rešitev mora biti enostavno dostopna vsem uporabnikom, brez nameščanja dodatnih komponent na uporabniški strani,
- uporabniki informacijske rešitve naj bodo vsi uporabniki navideznega zasebnega omrežja lizing podjetja,
- informacijska rešitev mora vsebovati mehanizem za določitev pravic dostopa do posameznega sklopa funkcionalnosti ter upoštevati ta dostop pri prikazu podatkov uporabniku,
- informacijska rešitev mora vsebovati mehanizem, da uporabniki posameznih odvisnih družb vidijo le podatke družbe, ki ji pripadajo, obenem pa se lahko določenim uporabnikom omogoči, da vidijo podatke vseh družb,
- informacijska rešitev naj vsebuje splošno področje, ki vsebuje splošne funkcionalnosti lizing podjetja, kot so: telefonski imenik, interni dokumenti ipd.,
- informacijska rešitev naj vsebuje področje prodaje, ki je namenjeno izvajanju opravil prodajalcem,
- informacijska rešitev naj vsebuje področje financ, kjer se nahaja orodje za spremljanje likvidnostnega tveganja,
- informacijska rešitev naj vsebuje področje administracije, ki je namenjeno skrbnikom sistema za upravljanje vseh skrbniških opravil nad uporabniki,
- informacijska rešitev naj bo izdelana tako, da je sistem enostavno razširiti z novimi področji ali novimi funkcionalnostmi področji skupaj z varnostnimi mehanizmi,
- informacijska rešitev naj vodi podroben seznam o aktivnostih uporabnika.

5.3.2 Uporabniki sistema

Podjetje ima vzpostavljeno navidezno zasebno omrežje z vsemi odvisnimi družbami. Do novega informacijskega sistema lahko dostopajo vsi uporabniki, ki so uporabniki navideznega zasebnega omrežja lizing podjetja. Zaradi velikega števila uporabnikov in lažjega skrbništva sistema mora sistem vsebovati več uporabniških skupin, v katere uvrščamo uporabnike. Glede na pripadnost ustrezni uporabniški skupini uporabnik pridobi ustrezne dostopne pravice do funkcionalnosti posameznih področij. Uporabniške skupine se kreirajo glede na potrebe dostopnih pravic po področjih.

5.3.3 Arhitektura sistema

Glede na opravljeni popis trenutnega stanja informacijskega sistema v podjetju, podan v poglavju 4.5, ter popis osnovnih zahtev poslovnega področja sem kot glavni razvijalec

sistema zasnoval arhitekturo novega sistema. Ta je večravenska in temelji na uporabi Microsoftovih tehnologij in orodij. Izbor slednjega je predvsem posledica že obstoječe sistemske arhitekture lizing podjetja in dejstva, da je imela razvojna ekipa prav z uporabo teh tehnologij največ izkušenj.

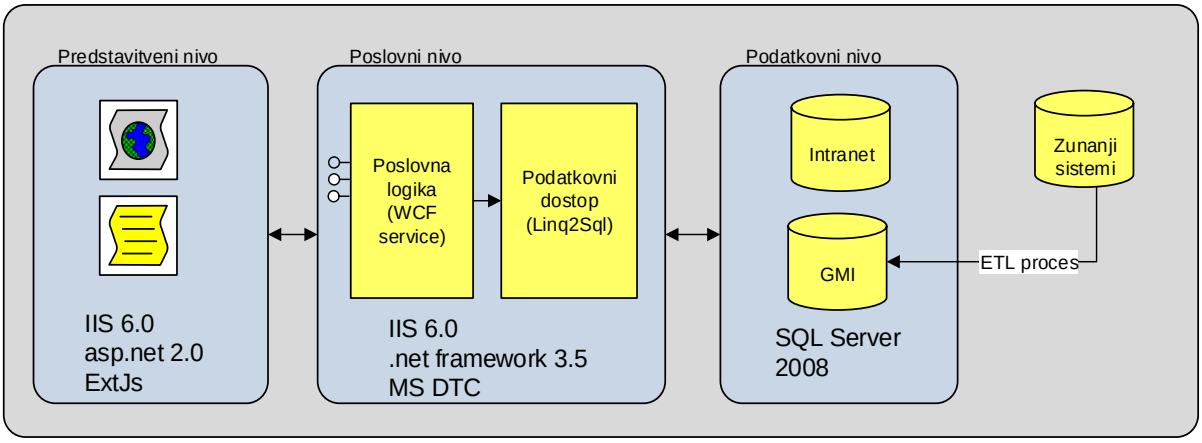
S pomočjo večravenske arhitekture smo želeli zgraditi sistem, ki ga bo enostavno nadgrajevati, dopolnjevati in vzdrževati. Prednosti take arhitekture informacijskega sistema izhajajo iz njenih lastnosti in so (ICommerce Design Strategies, 2012):

- enkapsulacija: vsaka raven skriva zapletenost svoje zgradbe pred ostalimi. Prehodi med ravnmi so določeni s programskimi vmesniki. Navedena lastnost izboljšuje prožnost in razširljivost sistema ter poenostavlja vzdrževanje. Posegi na posamezni ravni ne vplivajo na delovanje ostalih, če pri tem ne spreminjamo programskih vmesnikov, prek katerih so ravni medsebojno povezane,
- ločitev odgovornosti: sistem je lažje obvladljiv, ker vsaka raven združuje vsebinsko povezano množico funkcionalnosti. Napake so lokalizirane in omejene na posamezno raven. Padec zmogljivosti na posamezni ravni ne vpliva na delovanje ostalih ravni. V takem primeru je treba dodati sistemske vire samo na eni ravni, da se zmogljivost celotnega računalniškega sistema poveča v skladu s potrebami,
- ponovna uporaba izdelanih programskih rešitev. Posamezne programske rešitve lahko zaradi ločenosti ravni povezujemo na različne načine in tako razvijamo funkcionalnosti sistema,
- hitrejši razvoj novih funkcionalnosti. Funkcionalna ločenost ravni omogoča enostavnejše uvajanje novih programskih rešitev.

Večravenska arhitektura zagotavlja visoko raven varnosti, ki se zahteva pri najbolj občutljivih sistemih. To lahko zagotovimo z varnostnimi pravili na prehodih med posameznimi ravnmi. Med predstavitveno in poslovno raven na primer namestimo požarno pregrado, ki zagotavlja nadzor nad uporabo sistemskih virov in omejuje pravice uporabnikov skladno z veljavno varnostno politiko pri dostopu iz javnega omrežja. Občutljivi deli informacijskega sistema so tako varno skriti pred zlonamernimi obiskovalci, ne da bi bila funkcionalnost celotnega sistema zaradi tega kakorkoli okrnjena.

Večravenska arhitektura zagotavlja visoko skalabilnost in odpornost na napake. Pri velikem povečanju števila transakcij lahko obstoječe zmogljivosti enostavno razširimo tako, da vključimo dodatne sistemske vire na posameznih ravneh. Prednosti tega pristopa so v tem, da lahko selektivno širimo zmogljivosti brez nepotrebnega nadgrajevanja kapacitet, in v enostavnejšem in cenejšem vzdrževanju. Z dodajanjem redundantnih računalniških virov na posameznih ravneh lahko dodatno zavarujemo najbolj kritične dele sistema in odpravimo točke ogroženosti, ki lahko povzročijo izpad delovanja celotnega sistema.

Slika 5: Arhitektura novega informacijskega sistema



Vir: P. Starbek, Tehnična specifikacija, 2011, str. 7.

5.3.3.1 Predstavitvena raven

Predstavitvena raven prek odjemalca skrbi za interakcijo med uporabnikom in računalniškim sistemom. Na njej se izvajajo enostavne operacije, kot so preverjanje pravilnosti vnosa, prikazovanje podatkov in vnašanje podatkov v sistem. Poznamo več vrst odjemalcev; od najenostavnejših, kot je ukazna vrstica, do bolj kompleksnih, ki poleg zgoraj omenjenih operacij omogočajo tudi bolj kompleksne podatkovne obdelave in izpise.

V zadnjih letih se najbolj uveljavlja spletni odjemalec. Njegova prednost je predvsem, da s strani uporabnika ne zahteva nobenega dodatnega nameščanja na svoj računalnik, saj se za interakcijo uporablja običajni spletni brskalnik. Tako uporabnik lahko zelo enostavno dostopa do svoje aplikacije, obenem pa razpolaga vedno z najnovejšo različico želenega odjemalca.

Ker ima lizing podjetje hčerinska podjetja zunaj naših meja, služba za informatiko pa se nahaja le v matičnem podjetju, je bil to tudi glavni razlog, da smo se odločili za spletnega odjemalca.

Na predstavitveni ravni smo tako izbrali tehnologijo asp.net, ki je bila vodilna tehnologija razvoja spletnih aplikacij v Microsoft okolju, obenem pa je bila tehnologija, s katero smo imeli največ izkušenj. Omogoča razvoj dinamičnih spletnih strani in že vsebuje močne strežniške kontrole.

Asp.net ogrodje samo po sebi ne vsebuje dovolj naprednih gradnikov, ki bi omogočali enostavno komunikacijo s poslovno ravnjo ter prikaz tabelaričnih rezultatov, ki bi že podpirali standardne uporabniške operacije, kot sta sortiranje in filtriranje. V novo nastajajočem sistemu pa se je predvidevalo največ uporabniških interakcij prav tega tipa.

Zato smo se odločili za uporabo dodatne skriptne knjižnice ExtJs, ki odpravlja to pomanjkljivost ter s katero smo tudi že v preteklosti uspešno vključevali rešitve.

Za razvoj predstavitvene ravni se je uporabilo razvojno orodje Visual Studio 2010 in programski jezik C#.

5.3.3.2 Poslovna raven

Poslovna raven zagotavlja podporo za obdelovanje poslovnih podatkov po predpisanih poslovnih pravilih. Vsebujejo standarden nabor storitev in funkcij, ki so potrebne za zanesljivo, usklajeno delovanje celotnega sistema in dobro izkoriščenost sistemskih virov. Sistemske storitve poslovne ravni vsebujejo podporo za izvajanje programskih komponent, upravljanje s poslovnimi transakcijami in sejami, module za povezovanje z drugimi sistemi, upravljanje s skupnim fondom povezav na podatkovni strežnik, omrežne storitve, varnostne mehanizme, nadziranje izvajanja programske kode, obveščanje ob nastanku napak ali izpadih sistemskih virov ipd.

Poslovna raven nove rešitve sestavljata dve glavni komponenti:

- poslovna logika – se izvede kot spletna storitev, ki omogoča standardni SOAP-protokol. To omogoča enostavno povezovanje z vsemi odjemalci, ki podpirajo povezovanje s SOAP-protokolom spletnih storitev. Povezava predstavitvene in poslovne ravni tako ni problematična, saj Asp.net ogrodje ter ExtJs skriptna knjižnica, omenjena v opisu predstavitvene ravni, že sama po sebi podpirata protokol ter standarde za povezovanje s spletnimi storitvami SOAP. Za potrebe transakcij poslovna logika uporablja storitev MS DTC,
- podatkovni dostop – predstavlja komponento, ki skrbi za komunikacijo med poslovno in podatkovno ravnjo. Za enostavno implementacijo tega se uporabi tehnologijo LinqToSql, ki omogoča hitro in enostavno dodajanje ter spreminjanje dostopnih možnosti do posameznih objektov podatkovne ravni.

Tudi za razvoj poslovne ravni se je uporabljalo razvojno orodje Visual Studio 2010 in programski jezik C#.

5.3.3.3 Podatkovna raven

Podatkovna raven je namenjena trajnemu shranjevanju podatkov in stanj objektov, ki nastopajo pri obdelavi na poslovni ravni. Med sisteme, ki jih uvrščamo na podatkovno raven, prištevamo različne podatkovne baze, datotečne sisteme in obstoječe poslovne informacijske sisteme. Navadno vsebujejo vnaprej vgrajena pravila, ki preprečujejo nelogične spremembe podatkov. Komunikacija z zunanjimi sistemi poteka na osnovi

transakcij. V primeru, da med transakcijo pride do kakršnekoli napake, zna sistem sam vzpostaviti stanje pred začetkom transakcije.

Za podatkovni sistem smo se odločili, da se uporabi MS SQL Server 2008. Razloga za sprejem take odločitve sta:

- v podjetju že obstaja poslovni sistem, ki uporablja podatkovni strežnik MS SQL Server 2008,
- tehnologija poslovne ravni omogoča enostavno povezovanje na tip podatkovnega strežnika MS SQL Server.

Za potrebe novo nastajajočega sistema sta se določili dve podatkovni zbirki, kamor se shranjujejo podatki in so neposredno povezani z novo nastajajočim sistemom:

- **podatkovna zbirka Intranet**, ki je namenjena shranjevanju podatkov vseh novih funkcionalnosti, ki bodo nastale med razvojem,
- **podatkovna zbirka GMI,** ki služi kot podatkovno skladišče obstoječih informacijskih sistemov lizing podjetja. Novi sistem v omenjeno podatkovno zbirko ne zapisuje, ampak iz nje samo bere. Za ustreznost podatkov zbirke skrbi ETL-proces, ki zahtevane podatke iz drugih podatkovnih virov lizing podjetja vsakodnevno prenaša v omenjeno podatkovno zbirko.

5.3.4 Seznam zahtev

DSDM predlaga uporabo pristopa MoSCoW, ki sem ga podrobno opisal v poglavju 3.3.5, zato smo vse prepoznane zahteve prvega inkrementa postavili na prioritetni seznam. Pri tem smo se odločili, da v danem trenutku ne bomo določili zahtev, ki spadajo v zadnjo skupino omenjenega pristopa. To so zahteve, ki naj jih ne bi implementirali. Če se bo izkazalo, da ne moremo implementirati vseh zahtev tretje skupine – zahtev, ki bi jih lahko izvedli, bomo naredili premik takih zahtev v zadnjo skupino – zahteve, ki naj jih ne bi implementirali.

Pri določanju prednostih zahtev poslovnega področja se je upoštevalo predvsem odstotkovno višino zneska, ki ga denarni tok zavzame glede na skupen znesek vseh denarnih tokov. Tako so za lizing podjetje pravilno generirani amortizacijski načrti najetih posojil, pravilno vneseni podatki in ustrezna nastavitev parametrov ključni za učinkovito upravljanje z denarnimi tokovi, saj prek njih upravljajo kar z 900 milijoni EUR.

Tabela 3: Seznam zahtev po pristopu MoSCoW

Zahteve, ki jih nujno implementiramo	Zahteve, ki jih je dobro implementirati	Zahteve, ki jih lahko implementiramo
• prijava • odjava • kreiranje uporabnika • popravljanje uporabnika • revizijska sled aktivnosti uporabnika • onemogočenje uporabnika • urejanje dostopov do področij • urejanje pripadnosti uporabniškim skupinam • urejanje amortizacijskega načrta najetega posojila • urejanje najave prodajalca • urejanje najave odvisnih družb • urejanje obrestno izvedenih finančnih instrumentov • pregled denarnih tokov	• sprememba gesla • kreiranje uporabniških skupin • popravljanje uporabniških skupin • urejanje zadolževanj pri bankah • urejanje drugih denarnih tokov • urejanje amortizacijskega načrta revolving posojil • urejanje amortizacijskega načrta vezave denarnih sredstev	• pozabljeno geslo • kreiranje področij • popravljanje področij • urejanje korekcij zapadlih terjatev • urejanje operativnih stroškov po meri • pregled in iskanje telefonskega imenika podjetja • pregled in iskanje internih dokumentov • urejanje telefonskega imenika podjetja • urejanje internih dokumentov

5.3.5 Časovna okna

Časovno okno je izjemno pomemben pristop modela DSDM. S časovnimi okviri projekt v največji možni meri razdelimo na manjše dele ter tako omogočimo realizacijo rešitve v okviru fiksnega časa in stroškov. Edina preostala spremenljivka enačbe ostane »zahteve rešitve«, kar projektnemu vodji daje možnost opustitve najmanj pomembnih zahtev po pristopu MoSCOW (DSDM Public Version 4.2 Manual, 2008).

Projekt razvoja ogrodja in začetnih funkcionalnosti je imel določen rok trajanja sedem mesecev. To je bil tudi rok, ko je bilo treba s strani matične banke lizing podjetja zadostiti določenim regulativnim zahtevam. Te so po preteku obdobja sedmih mesecev zahtevale kvartalno oddajo različnih poročil, narejenih na osnovi stanja denarnih tokov. Tako se je

pričakovalo, da bo v obdobju sedmih mesecev projekt razvoja ogrodja in ključnih funkcionalnosti uspešno zaključen.

Temu sledi več inkrementalnih ciklov razvoja preostale potrebne funkcionalnosti v časovnih okvirih 2 mesecev.

V nadaljevanju predstavim opredelitev časovnih oken za razvoj ogrodja in začetnih funkcionalnosti sistema, ki je bil eden od ključnih korakov tudi za vse nadaljnje inkremente razvoja. Opredelitev ciklov za nadaljnje inkremente razvoja ni potrebna, saj je vsak nadaljnji inkrement že bil časovno opredeljen na čas dveh mesecev in ni vseboval dodatnih iteracij. Bolj podrobna obrazložitev je podana v nadaljevanju dela, v poglavju 8.

Za razvoj ogrodja in začetnih funkcionalnosti smo obdobje razdelili v več časovnih oken, kot kaže Tabela 4.

Tabela 4: Razporeditev zahtev ogrodja in ključnih funkcionalnosti

<table>
<tr><th>Mesec</th><th>Zahteve</th><th>Časovno okno</th></tr>
<tr><td>1</td><td rowspan="2">Začetno načrtovanje in oblikovanje
Za popolno razumevanje vseh zahtev in pravilno zasnovo se izvede več sestankov med vsebinskimi nosilci, v katere je vključen tudi razvijalec. Faza je izredno pomembna, da dobimo takoj čim bolj pravilno rešitev. Na osnovi pogovorov se izvedeta dva prototipa. Prvi prototip kaže primer računanja amortizacijskih načrtov danih posojil, ki je eden od najpomembnejših gradnikov rešitve. Naredi ga poslovni analitik v programskem paketu Excel. Prototip bo obenem pomoč razvijalcu rešitve za osvojitev kompleksne logike izračuna amortizacijskih načrtov. Drugi prototip napravi razvijalec informacijske rešitve. Prototip preizkusi predlagano arhitekturo sistema, izvede se zahteve za prijavo ter odjavo v sistem ter izvede prikaz enega področja z enostavnim tabelaričnem pregledom za testiranje uporabniške izkušnje. Prototipa se čim hitreje posredujeta v testiranje ključnim uporabnikom.
Končni izdelek okna: ER-diagram, diagram poteka, prototip izračuna amortizacijskega načrta v Excelu, začetni prototip rešitev.</td><td rowspan="2">1</td></tr>
<tr><td>2</td></tr>
<tr><td>3</td><td>Zahteve, ki jih je nujno implementirati
Ko temeljito poznamo zahteve, se lotimo vključevanja nujnih zahtev v prototipno rešitev. Pri snovanju sistema upoštevamo povratno informacijo testiranja prejšnjega časovnega okna in</td><td>2</td></tr>
</table>

4	ustrezno prototipno rešitev prilagodimo. **Končni izdelek okna:** prototip, vsebuje zahteve, ki jih je nujno implementirati.	
5	**Zahteve, ki jih je dobro implementirati** Vključimo zahteve, ki jih je dobro implementirati v prototip. Poleg tega zberemo povratne informacije od prototipa prejšnjega časovnega okna in opravimo vse potrebne spremembe. **Končni izdelek okna:** ER-diagram, diagram poteka, dopolnjen prototip, vsebuje zahteve, ki jih je dobro implementirati.	3
6	**Zahteve, ki jih lahko implementiramo** Vključimo zahteve, ki jih lahko implementiramo v prototip. Tudi tu upoštevamo vse povratne informacije preteklega prototipa in opravimo potrebne spremembe. **Končni izdelek okna:** ER-diagram, diagram poteka, dopolnjen prototip, ki vsebuje zahteve, ki jih lahko implementiramo.	4
7	**Pregled** Potreben za dokončno izvedbo in pregled vsega do sedaj opravljenega dela. To vključuje tudi izdelavo in dopolnitev dokumentacije ter postavitev sistema v produkcijsko okolje. **Končni izdelek okna:** dokončana rešitev prvega inkrementa, končna dokumentacija.	5

Kot je razvidno iz tabele 4, ki prikazuje razporeditev zahtev po časovnih oknih, sta prvo in drugo časovno okno časovno obsežnejša. Razlog za to je v tem, da smo dali poudarek na dobro zasnovo rešitve, ki je ključna za vse nadaljnje delo. Logika, ki jo rešitev vsebuje, je kompleksnejša, zato smo morali najprej s prototipiranjem dobro preveriti vse postopke izračuna kot tudi zastavljeno arhitekturo sistema in uporabniško izkušnjo pri ključnih uporabnikih. Po našem mnenju sta bili prvi dve časovni okni ključni za uspešno dokončanje rešitve, zato smo jim tudi namenili nekoliko več časa.

5.4 Iteracija funkcionalnega modela

Faza za vsako iteracijo vključuje analizo zahtev, katere ključna izida sta funkcionalen prototip in seznam analize funkcionalnosti.

Zahteve prioritetnega seznama, ki jih obravnavamo v dani iteraciji, smo najprej združili v funkcionalno logične skupine. Za ilustracijo podajam primer logične združitve zahtev, ki jih nujno implementiramo in so del drugega časovnega okna.

Tabela 5: Primer logične združitve zahtev

Administracija	• prijava • odjava • kreiranje uporabnika • popravljanje uporabnika • revizijska sled aktivnosti uporabnika • onemogočenje uporabnika • urejanje dostopov do področij • urejanje pripadnosti uporabniškim skupinam
Amortizacijski načrt najetega posojila	• urejanje amortizacijskega načrta najetega posojila
Prodaja	• urejanje najave prodajalca • urejanje najave odvisnih družb
Obrestno izvedeni finančni instrumenti	• urejanje obrestno izvedenih finančnih instrumentov
Denarni tokovi	• pregled denarnih tokov

Vsako izmed tako dobljenih skupin smo nato podrobno analizirali in funkcionalnosti zapisali v podroben seznam funkcionalnih zahtev, ki jih v naslednji fazi tudi implementiramo. Funkcionalne zahteve so tekstovni dokumenti, ki predstavljajo funkcionalnost sistema skozi cilje in perspektive uporabnika. Opisane so tako podrobno, da na osnovi tega lahko načrtamo in zgradimo sistem, ki bo v največji meri zadovoljil potrebe uporabnikov. Osnovni namen seznama funkcionalnih zahtev sta učinkovito zajemanje in predstavitev zahtev uporabnikov. Pri njihovem snovanju nas zanimajo predvsem odgovori na vprašanja, kot so:

1. kdo uporablja sistem,
2. kakšni so njihovi tipični scenariji in
3. kakšni so njihovi cilji uporabe.

V fazi testiranja nam tako zapisane funkcionalne zahteve nato služijo kot ključni dokument za izdelavo testnih scenarijev in preverjanje ustreznosti delovanja izdelanega sistema. Primer seznama analize funkcionalnih zahtev podajam na Sliki 6.

Slika 6: Izsek iz seznama funkcionalnih zahtev za logično skupino administracije

Administracija

ADM-1	Prijava
Prijavi uporabnika v sistem z uporabniškim imenom in geslom. Uporabnik pripada ustrezni uporabniški skupini. To je zapisano v piškotek, ki se uporablja ves čas seje. Različne skupine uporabnikov imajo različne pravice v zvezi z področji. **Utemeljitev**: Če želite prepoznati različne skupine uporabnikov za različne funkcije privilegiranih	

ADM-2	Odjava
Odjavi uporabnika iz sistema. Prav tako zaradi varnosti izbriše vse podatke seje ter piškotke. Tudi če se uporabnik ne odjavi, se seja samodejno prekine čez nekaj časa. **Utemeljitev**: Za zagotovitev načina za izbris varnostno občutljivih podatkov.	

ADM-3	Kreiranje uporabnika
Skrbnik sistema ima možnost kreirati novega uporabnika sistema. Vnesti mora uporabniško ime navideznega omrežja leasing podjetja zato mora uporabnik prej obstajat v navideznem omrežju leasing podjetja. **Utemeljitev**: Za zagotovitev vnosa sprememb podatkov uporabnika v sistem.	

ADM-4	Popravljanje uporabnika
Skrbnik sistema ima možnost popravljanja podatkov uporabnika sistema. Ne more popraviti uporabniškega imena na katerega je uporabnik povezan. **Utemeljitev**: Za zagotovitev pravilnih podatkov uporabnika v sistemu.	

ADM-5	Onemogočenje uporabnika
Skrbnik sistema ima možnost za določen čas onemogočiti uporabnika. V času onemogočenja uporabnik obstaja v sistemu vendar sam ne more dostopati do sistema. **Utemeljitev**: Za zagotovitev varnostnih pravil podjetja.	

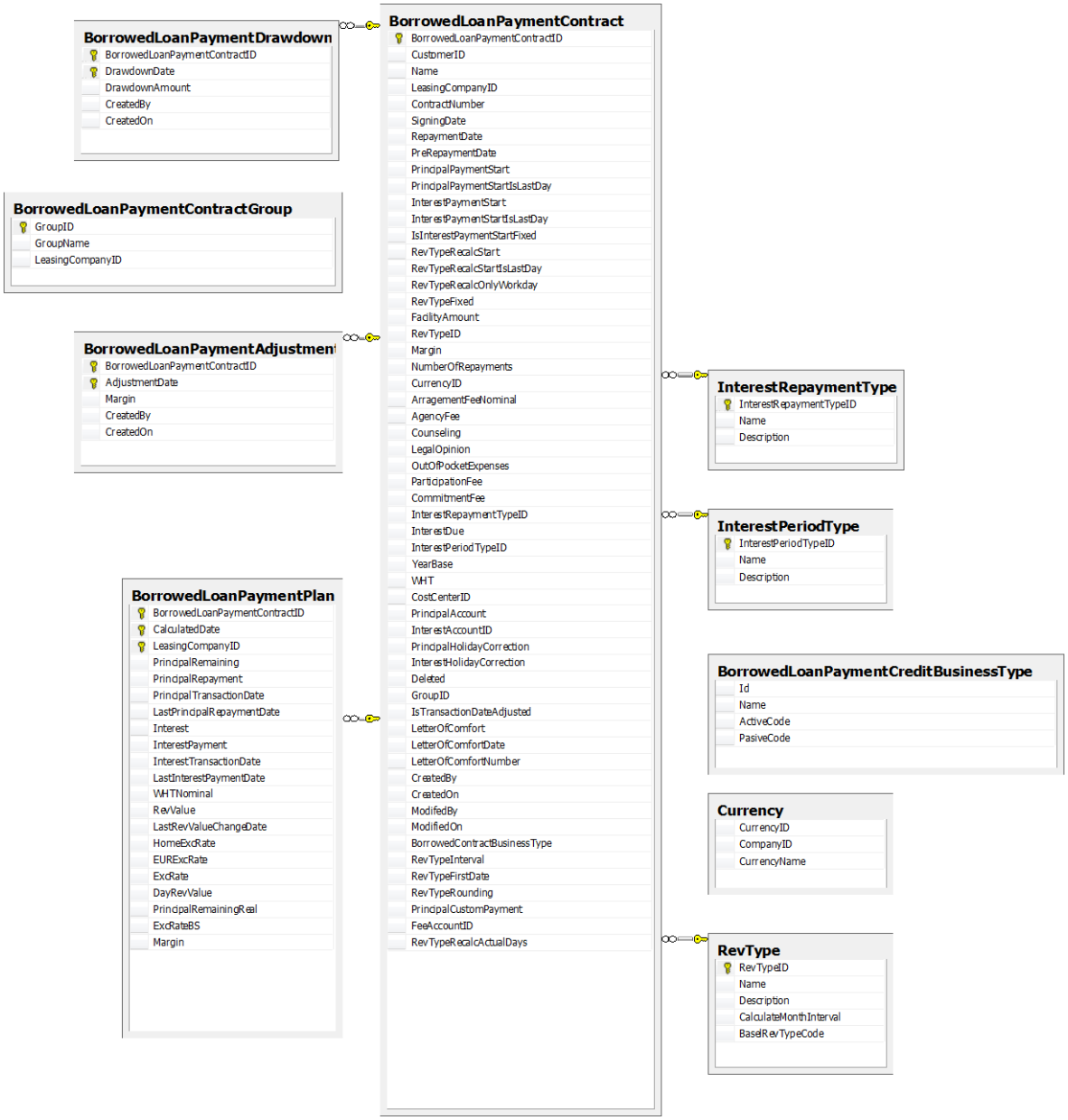
Vir: Lizing podjetje, interna dokumentacija, 2011.

5.5 Iteracija načrtovanja in izgradnje

Ključna naloga faze je vgradnja seznama funkcijskih zahtev prejšnje faze v sistem na način, da čim bolj zadovoljimo potrebe uporabnika. V tej fazi rešitev zgradimo s pomočjo načrtovanja, funkcionalnih prototipov ter na pripombah in predlogih uporabnikov.

Za predvidene zahteve smo najprej izdelali entitetno-relacijski podatkovni model zahtev trenutne iteracije. Iz seznama zahtev smo najprej identificirali potrebne tabele in določili primarne ključe, s čimer onemogočimo podvajanje zapisov. Nato smo tabele med seboj povezali prek tujih ključev in za mnogo relacij načrtali vmesne tabele. Tako kot zahteve so bili tudi podatkovni modeli načrtovani po vsebinsko povezanih sklopih. Primer enega izmed ključnih entitetno-relacijskih modelov prikazuje Slika 7.

Slika 7: Entitetno-relacijski model za amortizacijske načrte najetega posojila



Vir: Lizing podjetje, Tehnična dokumentacija, 2011.

Vsakemu entitetnemu modelu je sledil podrobnejši opis polj tabel, predstavljen na Sliki 8.

Slika 8: Izsek iz podrobnega opisa polj tabel entitetno-relacijskega modela za amortizacijske načrte najetega posojila

Ime podatka	Obvezen	Opis
BorrowedLoanPaymentContractID	Da	Identifikator pogodbe
CustomerID	Da	Identifikator banke pri kateri se najema kredit. Povezava na tabelo Customer v GMI zbirki glede na podani LeasingCompanyID zapisa.
Name	Da	Ime kreditne pogodbe
LeasingCompanyID	Da	Identifikator družbe znotraj skupine NLB Leasing, ki najema kredit. Povezava na tabelo **Company**.
ContractNumber	Da	Stevilka pogodbe.
SigningDate	Da	Datum sklenitve pogodbe.
RepaymentDate	Da	Datum poplačila pogodbe.

Vir: Lizing podjetje, Tehnična dokumentacijam 2011,

5.6 Implementacija

Fazi načrtovanja sledi faza implementacije, v kateri razvojni tim z vsemi informacijami vseh predhodnih faz poskuša razviti rešitev. Ob koncu faze se testirana rešitev iteracije izroči uporabnikom, projekt pa se vrne v naslednjo iteracijo funkcionalnega modela. Če je treba, se izvede tudi šolanje uporabnikov.

5.6.1 Razvojno okolje

Glede na predlagano arhitekturo, opisano v točki 5.3.3, ter izkušnje razvijalcev smo se odločili za uporabo naslednjih tehnologij in orodij:

- Asp.net 3.5,
- Microsoft Sql Server 2008,
- Microsoft Interenet Information Server (IIS) 7.0,
- Visual Studio 2010 in
- Sql Server Enterprise Manager.

5.6.2 Stil programiranja in pravila razvoja

Ko govorimo o **stilu programiranja,** mislimo na način pisanja izvorne kode. Program se lahko izvaja tudi, če je napisan v eni vrstici, vendar se pisanje izvorne kode na ta način šteje kot slab zaradi slabe berljivosti kode. Pomembno je tudi dokumentiranje metod, opazk in pojasnil znotraj metod, ki predstavljajo glavni izvor podatkov za tehnično dokumentacijo rešitve.

Stil programiranja in možnosti pri zagotavljanju njegove visoke kakovosti nam določa tudi uporabljen programski jezik. Ta je v veliki odvisnosti od uporabljenih tehnologij in orodij. Glede na to smo se odločili za uporabo objektno orientiranega programskega jezika C#. Ta je zelo priljubljen programski jezik razvoja rešitev, ki temeljijo na tehnologijah Microsoft. Zagotavlja nam dovolj veliko enostavnost in hkrati širino ter prilagodljivost pri njegovi uporabi. Poleg objektnega jezika smo se na ravni podatkovne baze odločili za uporabo TSQL-jezika. Ker je poslovna logika močno podatkovno orientirana in smo jo hkrati želeli imeti dovolj prilagodljivo, se vso tako logiko izvede neposredno na podatkovnem strežniku z uporabo programskega jezika TSQL.

S stilom programiranja so močno povezana tudi **pravila razvoja**, ki jih sodelujoči na projektu uskladijo pred začetkom izvajanja ter so po navadi rezultat izkušenj uporabe različnih dobrih praks posameznih sodelujočih. Pravila razvoja so nujno potrebna, ker samo z njimi izdelana koda postane:

- ponovno uporabna,
- vzdrževana in
- standardizirana.

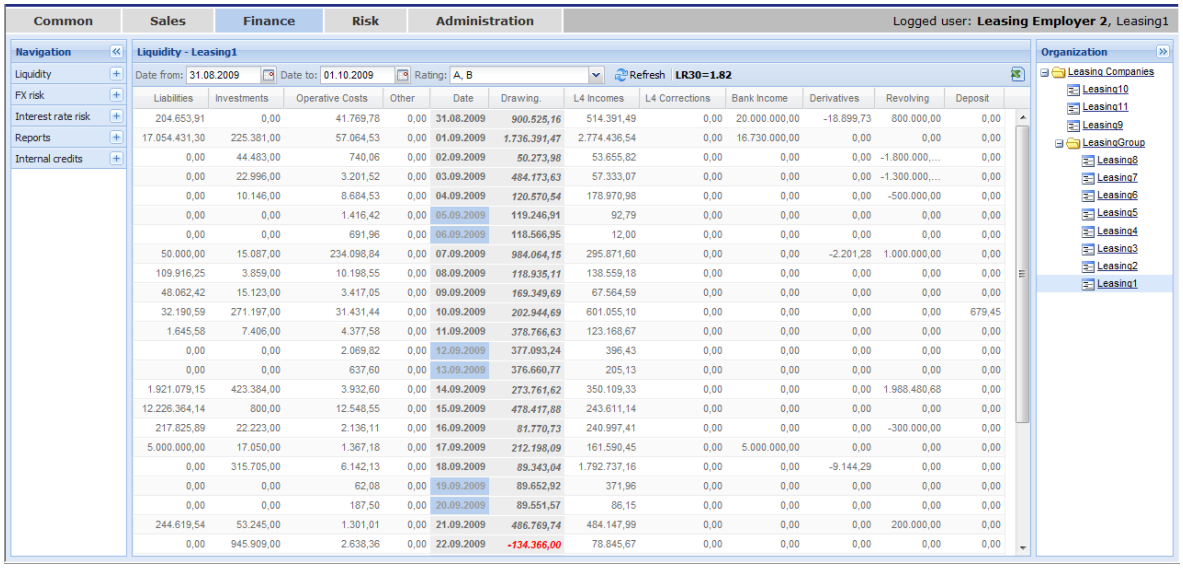
V ekipi smo se med člani razvojnega tima dogovorili in sprejeli naslednja pravila, s katerimi močno poenostavimo vključevanje novih članov v projekt ter kasneje tudi vzdrževanje projekta:

- **enostavnost** – procesi naj bodo izdelani z minimalno količino kode. Če je le mogoče, se izogibamo dolgih metod. Če metoda postane predolga, jo je treba razbiti v manjše dele z uporabo podmetod,
- **opisnost** – ob pisanju kode naj bo vedno prisotno zavedanje, da je lahko koda dana v pregled ostalim članom. Kljub temu da je nemogoče vedno zagotoviti, da je koda vedno razumljiva na prvi pogled, mora biti napisana tako enostavno, da ne zahteva vedno ponovnega raziskovanja. Pri poimenovanju spremenljivk in metod je treba upoštevati, da so dovolj opisne in obenem dovolj kratke,
- **berljivost** – uporablja naj se dovolj komentiranja in zamikanje vrstic, hkrati pa se izogiba slabim nerazumljivim gnezdenjem. Pred klici metod ali znotraj glavnih postopkov so obvezni kratki komentarji. Ti nato delujejo kot opombe za vsakogar, če je potrebno posodabljanje,
- **pravila pisanja kode** – uporabljamo angleške nazive spremenljivk in metod. Vse javne metode in spremenljivke se pišejo z veliko začetnico, medtem ko se zasebne z malo začetnico. Pri metodah in spremenljivkah, sestavljenih iz več besed kot separator, uporabljamo veliko začetnico,
- **preurejanje kode** – vsakdo ima pravico, da spremeni del kode, če meni, da jo lahko izboljša. Če gre za večji poseg, se to naredi s posvetom članov razvojnega tima, ki

potrdi primernost sprememb. S tem se zelo povečajo možnosti odprave slabih delov kode, ki bi v nasprotnem primeru ostale neopažene,

- **oznake glav kode** – vsaka javna metoda naj ima oznako glave, ki vsebuje namen, opis vhodnih spremenljivk, opis izhodnih spremenljivk, avtorja in zadnji datum popravka. V primeru, da nekdo popravi metodo, mora popraviti tudi oznako glave,
- **preverjanje ustreznosti podatkov** – enostavna preverjanja ustreznosti podatkov izvajajo na strani odjemalca (JavaScript). Na strani strežnika izvajajo le preverjanja, ki zahtevajo preverjanje z dostopom do podatkovne baze,
- **načrtovanje** – vedno pred izvedbo je treba prediskutirati način izvedbe glavnih postopkov s člani tima. Podati jim je treba tako podroben opis postopka izvedbe, da ga lahko razumejo. S tem lahko ostali člani odkrijejo kakšne pomanjkljivosti še pred izvedbo ter s tem prihranimo čas kasnejšega popravljanja. Hkrati se s pregledom nad okvirnim delom drugega prek vseh članov tudi zmanjša verjetnost napak v metodah, ki jih izvaja eden in kliče drugi član tima,
- **testiranje** – uporaba in zagotavljanje enot testiranja za dele kode v poslovni logiki, ki predstavlja glavne postopke rešitve. Tako že v fazi razvoja zagotovimo lažje kasnejše izvajanje sprememb postopkov in enostavnejše vzdrževanje kode.

Slika 9: Zaslonska maska razvite rešitve za upravljanje z likvidnostnim tveganjem



5.6.3 Testiranje

Testiranje je ena izmed pomembnih faz razvoja programskih rešitev. Njegov glavni namen je zmanjševanje tveganja napačnega delovanja kot tudi zagotavljanje kakovosti, dostopnosti in pravilnosti delovanja rešitve. Pomembno je, da ob vsakokratnem preverjanju ugotovimo (Jenkins, 2008, str. 8):

1. Ali programsko opremo gradimo pravilno? Ta postopek imenujemo verifikacija.

2. Ali programska oprema ustreza osnovnim zahtevam? Ta postopek imenujemo validacija.

Z njegovim izvajanjem moramo pričeti v čim zgodnejši fazi, saj le takrat najhitreje in najenostavneje odpravimo napake.

Metode testiranja delimo v dve osnovni skupini (Jenkins, 2008, str. 9):

- **metode črne skrinjice ali funkcionalna analiza** – testiranje, ki ignorira notranje mehanizme sistema, ter se osredotoči na rezultate izhodov sistema, ki so posledica kakršnegakoli vhoda v sistem ter njihovega procesiranja,
- **metode bele skrinjice ali strukturna analiza** – testiranje, ki se osredotoča na notranje mehanizme sistema, kot sta koda in logika procesiranja vhodov.

Obstaja veliko vrst testiranj, pri katerih vsaka vrsta zagotavlja določeno stopnjo kakovosti končne izgrajene rešitve. Med njimi najpogosteje srečamo (Jenkins, 2008, str. 9–14):

- **test enot**: je testiranje posameznih enot ali skupin povezanih enot. Največkrat ga izvaja programer, v katerem preveri, da za dane vhodne podatke dobimo ustrezne pričakovane izhodne podatke,
- **integracijsko testiranje**: je test, kjer preverimo, da posamezni programski deli sistema med seboj pravilno sodelujejo. S tem potrdimo ustrezno povezljivost med deli sistema,
- **funkcionalno testiranje**: je testiranje, kjer se prepričamo, da navedena funkcionalnost, definirana v zahtevah, resnično deluje,
- **sistemsko testiranje**: s sistemskim testiranje se prepričamo, da rešitev deluje v različnih sistemskih okoljih,
- **obremenitveno testiranje**: z njim testiramo delovanje rešitve v primeru izjemnih pogojev – zelo velika obremenitev sistema,
- **performančno testiranje**: z njim preverimo hitrost ter učinkovitost rešitve in se tako prepričamo, da do rezultatov pridemo v želenem času,
- **uporabnostno testiranje**: z njim preverimo uporabnost in prijaznost rešitve s stališča uporabniškega vmesnika,
- **sprejemno testiranje**: običajno ga izvaja testni uporabnik, s čemer se prepriča, da rešitev deluje v skladu z definiranimi zahtevami,
- **regresijsko testiranje**: je test po izvedbi sprememb rešitve ali delov rešitve, da se prepričamo, da spremembe delujejo pravilno ter da niso pokvarile obstoječega delovanja rešitve,
- **beta testiranje**: je testiranje končnega uporabnika tik pred pravo uporabo sistema, s katerim se odpravlja nepredvidljive napake.

Glede na pomembnost vsebovanosti posamezne kakovosti v končni rešitvi ustrezno vključimo deleže vrst različnih testiranj v rešitev.

Za kakovostno testiranje je treba pripraviti načrt testiranja. Z njim opredelimo, kako se bo testiranje izvajalo, testne primere, določimo vse udeležence testiranja, vrste testiranj ter predvidimo postopek dokumentiranja in sporočanja napak (Jenkins, 2008, str. 20–23).

Testiranje začnemo pri posameznih modulih, ki jih testiramo po principu bele skrinjice. Testirane module nato postopoma združujemo ali integriramo. Ta del testiranj običajno zagotavlja razvijalec sam. Ko je rešitev sestavljena, pred namestitvijo v produkcijsko okolje preverimo še skladnost funkcionalnosti iteracije z definiranimi zahtevami. To imenujemo sprejemno testiranje. Temelji na testnih primerih, definiranih v načrtu testiranja, ki jih izvajajo posebni testni uporabniki ter tako preverijo končno ustreznost delovanja. Ko je ta del testiranja uspešno končan, opravimo še razna sistemska testiranja in rešitev prenesemo v produkcijsko okolje.

Testni primeri so dogodki, ki jih želimo v fazi testiranja simulirati na razviti rešitvi. Pri pripravi testnih primerov izhajamo iz funkcionalne specifikacije. Ker je čas testiranja navadno omejen in tako ne moremo pokriti vseh možnih kombinacij testnih primerov, moramo biti pri izboru testnih primerov previdni in natančni. Pri tem na začetku upoštevamo nepisano pravilo, da poskušamo pokriti predvsem testne primere, za katere menimo, da jih bodo uporabniki največkrat uporabljali.

Slika 10: Izsek iz dokumenta testnih primerov – testni primer za prijavo uporabnika

T-ADM-1	**Prijava uporabnika**			
Namen: 1. Prijava kot "Uporabnik". **Pred testni pogoji:** za uporabnika z onemogočenim uporabniškim imenom uporabljaj "UporabnikOnem" za uporabnika s pretečenim geslom uporabljaj "UporabnikPret" **Preverjanje:** Da				
Št.	**Akcije**	**Pričakovan rezultat**	**Dobljen rezultat**	**Uspešnost**
1	Prijava z vsemi zahtevanimi informacijami	Prijava je uspešna in uporabnik je prijavljen v sistem	Prijavil se v sistem nahajam na prvi strani aplikacije	Uspešen
Preverjanje				
2	Prijava brez vseh potrebnih informacij	Prijava bo neuspešna če so naslednja polja prazna: a. Uporabniško ime b. Geslo Prikazana bodo naslednja sporočila v primeru da so zgoraj našteta polja prazna: a. Uporabniško ime ne sme biti prazno b. Geslo ne sme biti prazno	Ko sem se poskušal prijaviti brez zahtevanih polj, sem dobil ustrezna sporočila napak.	Uspešen
3	Prijava z napačnim uporabniškim imenom ali geslom	Uporabniku se prikaže sporočilo napake "Podano uporabniško ime ali geslo ni pravilno.".	Ob napačnem uporabniškem imenu ali geslu sem dobil napako.	Uspešen
4	Prijava z onemogočenim uporabniškim imenom	Uporabniku se prikaže sporočilo napake "Vaše uporabniško ime je preteklo. Prosim obrnite se na skrbnika sistema.".	Ko se poskusim prijaviti z onemogočenem uporabnikom dobim sporočilo.	Uspešen
5	Prijava s pretečenim geslom	Uporabniku prikaže obrazec za spremembo gesla.	Ko se prijavim z uporabnikom ki ima pretečeno geslo dobim obrazec za spremembo gesla.	Uspešen
6	Klik na pozabljeno geslo	Uporabnik je preusmerjen na obrazec za pozabljeno geslo.	Ob kliku na pozabljeno geslo dobim obrazec za pozabljeno geslo.	Uspešen
Opazke / Aktivnosti po testu: Nič				

Vir: Lizing podjetje, Tehnična dokumentacija, 2011.

Na koncu sprejemnega testiranja morajo uporabniki, udeleženi v testiranju, pripraviti poročilo in tako podati odločitev o ustreznosti razvite rešitve. Če testiranje ni ustrezno, se razvoj rešitve vrne nazaj v izvedbo. Tako se odpravi glavne pomanjkljivosti rešitve in nato preveri v ponovnem testiranju. Ko je testiranje uspešno zaključeno, se lahko nadaljuje zaključni del implementacije sistema – to je prenos izvedene funkcionalnosti iteracije sistema v produkcijsko okolje. Nato se projekt vrne v naslednjo iteracijo funkcionalnega modela. Če smo razvili vse funkcionalne modele, fazi sledi poprojektna faza.

5.7 Poprojektna faza

Poprojektna faza vsebuje dve glavni aktivnosti razvoja rešitve, predstavljeni v nadaljevanju.

5.7.1 Vzdrževanje

S trenutkom, ko izdelano rešitev ali del rešitve namestimo v produkcijo in je na voljo uporabnikom, se pojavi zahteva po njenem vzdrževanju. Čeprav rešitev ustrezno deluje, jo je treba nenehno prilagajati različnim zunanjim spremembam, od sprememb v zakonodaji do širitve poslovanja ter podpore novim procesom.

Vzdrževanja programske opreme bi lahko glede na njen izvor razdelili na štiri glavne aktivnosti (Solina, 1997, str. 192):

1. **popravki za odpravljanje napak**,
2. **prilagajanje programske opreme spremembam** v okolju, na drugo vrsto strojne ali sistemske programske opreme. Pri tem se funkcionalnost programske opreme ne spremeni,
3. **razširitev zmogljivosti programske opreme** se ukvarja z novimi ali s spremenjenimi zahtevami. Spremeniti moramo funkcije sistema, izboljšati zmogljivost sistema ali uporabniški vmesnik,
4. **preventivno vzdrževanje programske opreme** je namenjeno spremembam, ki izboljšajo možnost vzdrževanja (dopolnjevanje dokumentacije, izboljšava programske strukture itd.).

Pomembno je, da določimo natančen postopek vzdrževanja (Solina, 1997, str. 197–198):

1. vzdrževanje programske opreme moramo ustrezno organizirati, da ne bi prišlo do neavtoriziranih ali nasprotujočih sprememb programske opreme,
2. vsak zahtevek za vzdrževanje mora biti skrbno dokumentiran. V primeru napake moramo dokumentirati okoliščine, ki so pripeljale do napake,
3. zahtevke moramo preveriti in odobriti le tiste, ki so potrebni, smiselni in ekonomsko upravičeni,

4. glede na vrsto vzdrževanja sprožimo ustrezen postopek. V primeru kritične napake, ki zaustavi normalno delovanje neke organizacije, moramo takoj ukrepati z vsemi razpoložljivimi silami. Če gre za zadevo, ki lahko počaka, jo uvrstimo na seznam napak, ki jih rešujemo po vrsti na ustaljen način,
5. z organizacijskega vidika obravnavamo vsak zahtevek podobno kot razvoj nove programske opreme. Zahtevo za vzdrževanje moramo skrbno analizirati, spremembo ali popravek načrtovati in na koncu programsko opremo testirati.

Z končanjem ogrodja in glavnih funkcionalnosti projekta smo v proučevanem projektu prešli v obdobje vzdrževanja in razvoja funkcionalnih nadgradenj rešitve dvomesečnih ciklov. Slednji glede na sprejeti način dela skrbijo za redno dopolnjevanje funkcionalnih potreb rešitve. Vsak cikel obravnavamo kot svoj podprojekt, ki ga tudi naprej izvajamo po sprejeti metodologiji dela DSDM.

5.7.2 Analiza projekta

Po zaključku vsakega razvojnega projekta je treba analizirati, kaj v projektu je bilo opravljeno ustrezno in v katerih korakih bi lahko razvojni proces izboljšali. Pogledamo, kako uporaba izbrane metodologije ustreza načinu dela udeležencev v projektu, ter identificiramo točke, ki jih pri izvajanju postopkov razvoja lahko izboljšamo, ter tako zagotovimo optimalnejši razvojni proces za uporabo v prihodnjih projektih.

Vpliv trenutno uporabljene metodologije razvoja na uspešnost projekta je večplasten. Delimo ga lahko na merljivi – objektivni vpliv in nemerljivi oziroma težje merljivi – subjektivni vpliv.

Merljiv vpliv ugotovimo na podlagi številčne analize podatkov projekta. Ta zajema podatke, kot so začetek projekta, rok za zaključek projekta, vrednost projekta in dejanski zaključek projekta.

Težje merljivi vplivi na uspešnost projekta pa so zadovoljstvo stranke z izvedbo projekta in zadovoljstvo uporabnikov. Tega merimo prek standardizirane ankete, ki jo stranka izpolni po izvedbi vsakega projekta, ter prek intervjujev vseh udeleženih v projektu.

Po koncu vsakega projekta izdelamo tudi listo glavnih problemov, s katerimi smo se srečali med razvojem rešitve. Tudi ta nam je v veliki meri v pomoč pri snovanju in razvoju prihodnjih projektov. Primer prikazuje Tabela 6.

Tabela 6: Lista glavnih problemov v času razvoja sistema – prvi inkrement

Faza	Problem	Rešitev
Implementacija	Problem časovno potratnih podatkovnih obdelav za prikaz povzetih podatkov.	Ustrezna prilagoditev podatkov podatkovnega skladišča. Ustrezna postavitev indeksov.
Iteracija funkcionalnega modela	Hitre in nenadne spremembe zahtev matične banke po poročilih	Dovolj kratki inkrementi, da se lažje prilagajamo zahtevam. Pred implementacijo dejanske zahteve izvajalec še enkrat preveri ustreznost zahteve.
Študija izvedljivosti	Težko začetno usklajevanje med naročnikom in zunanjim izvajalcem zaradi prostorske dislociranosti	Izvajanje na lokaciji naročnika.
Testiranje	Prezasedenost uporabnikov za izvedbo ustreznega uporabniškega testa.	Načrtovanje posebnih terminov na ravni podjetja za skupinsko testiranje funkcionalnosti, vodeno s strani nadrejenega.

Vir: Lizing podjetje, Tehnična dokumentacija, 2011.

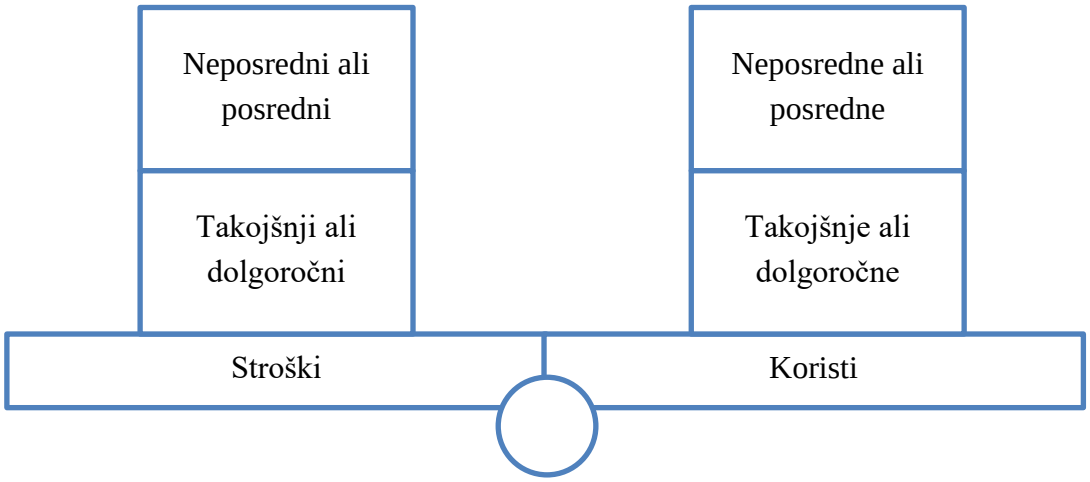
Podrobnejšo analizo uporabe metodologije DSDM in njenih zaznanih problemov v času proučevanega projekta podajam v poglavju 7 v nadaljevanju iz razloga, ker je ta eden izmed ključnih ciljev magistrskega dela.

6 ANALIZA EKONOMSKEGA VIDIKA NOVE REŠITVE

Investicije so v razvoju in rasti organizacije odločujoč dejavnik, zaradi česar njihova ocena predstavlja eno izmed najpomembnejših področij poslovnega odločanja. Pred pričetkom razvoja informacijskega sistema v organizaciji je tako treba opraviti podrobno analizo učinkov in tveganj, ki bolj podrobno opredeljuje, kaj mora biti narejeno, zakaj, v kakšnem časovnem obdobju in s kakšnimi stroški. Pri tem si pomagamo z različnimi metodami. Tako pridobljena analiza nam da temelje za odločitev za projekt in kasneje za spremljanje ter kontrolo faz projekta, saj lahko v vsakem trenutku preverimo, ali izvajanje projekta še vedno poteka znotraj časovnih, stroškovnih in funkcionalnih okvirjev.

Analiza stroškov in koristi (angl. *Cost Benefit Analysis*) je enostavna in široko uporabljena metoda, ki ocenjuje ter sešteva ekvivalentno denarno izražene stroške ter koristi investicijskega projekta in podaja mnenje o njegovi (ne)upravičenosti. Slika 11 prikazuje, da lahko imamo tako posredne kot tudi neposredne stroške in koristi. Ti lahko nastopijo takoj ali dolgoročno, ključno pa je, da jih ohranjamo v ravnotežju (Laudon, 2013, str. 567).

Slika 11: Vrste stroškov/koristi



Vir: K. C. Laudon, Management Information Systems - Managing the Digital Firm, 2013.

Neposredne koristi je mogoče količinsko kot tudi vrednostno opredeliti. Posredne koristi so težje merljive in so največkrat povezane z organizacijo in ljudmi. Običajno se jih v trenutku uvedbe sistema ne zavedamo, saj nastajajo dolgoročno (Laudon, 2013, str. 567).

Organizacija vlaga v informacijske sisteme za dosego naslednjih šestih ključnih koristi (Laudon, 2013, str. 567–568):

- **operativna odličnost**: učinkovitost, produktivnost in izboljšanje v poslovnih praksah ter upravljanju,
- **novi izdelki, storitve in poslovni modeli**: poslovni model opisuje, kako podjetje proizvaja, dobavlja in prodaja izdelka ali storitve. Informacijski sistemi in tehnologije ustvarjajo nove priložnosti za izdelke ter storitve in nove načine udejstvovanja v poslu,
- **zaupanje med kupci in dobavitelji**: boljša komunikacija in storitve povečujejo prihodke in znižujejo stroške,
- **izboljšano odločanje**: brez točnih in pravočasnih informacij vodstvo sprejema odločitve na podlagi napovedi ugibanj in sreče. Vse to velikokrat zvišuje stroške in vodi v izgubo kupcev,
- **konkurenčna prednost**: izvedba učinkovitih informacijskih sistemov omogoča cenejše in boljše izdelke ter s tem večjo prodajo ter dobičke od konkurence,
- **preživetje**: informacijski sistem lahko predstavlja tudi nujnost za poslovanje. To se največkrat odraža kot posledica zakonskih regulativ države, ki zahteva hranjenje zgodovine poslovanja in poročanja različnih informacij.

Glavne koristi finančne institucije po uvedbi sistema so naslednje (Hajne, 2011, str. 76–80):

- **zmanjšanje transakcijskih stroškov**: z implementacijo informacijskega sistema za upravljanje z nekreditnimi tveganji se je poleg učinkovitosti upravljanja teh tveganj povečala tudi stroškovna učinkovitost z vidika transakcijskih stroškov,
- **povečanje produktivnosti**: z uvedbo informacijskega sistema smo z uporabo sodobnih računalniških orodij ter z avtomatizacijo obdelave podatkov bistveno povečali produktivnost zaposlenih na finančnem področju,
- **učinkovitejša spremljava tveganj odvisnih družb**: orodje za upravljanje z nekreditnimi tveganji je omogočilo cenejšo, kakovostnejšo ter hitrejšo spremljavo strukture in višine stroškov, načrtovanih naložb, naložbenega portfelja in odplačevanja obveznosti odvisnih družb,
- **zmanjšanje kapitalskih rezerv iz naslova nekreditnih tveganj**: z uporabo orodja se je zmanjšalo tako tržno kot operativno tveganje, ki je vključeno v imenovalec izračuna kapitalske ustreznosti.

Za analizo finančnih posledic projekta razvoja informacijskega sistema lahko uporabimo številne metode. Priporočljiva je uporaba čim več različnih metod presoje donosnosti, saj več različnih metod poveča objektivnost presojanja investicijskih projektov in hkrati pomeni osvetlitev stanja iz različnih zornih kotov. V zasnovi in izvedbi sistema sem se omejil na štiri metode: dobo vračanja investicije, neto sedanjo vrednost, notranjo stopnjo donosa ter popravljeno notranjo stopnjo donosa (Čibej, 2006, str. 5).

Doba vračanja investicije je najenostavnejša metoda. Predstavlja čas, običajno izražen v letih in mesecih, ki je potreben, da se povrne začetni strošek. Pri njej sem upošteval naslednja dejstva, ki sem jih lahko izračunal na podlagi materialnih dejstev preteklih tokov:

- investicija v prenovo in dopolnitev informacijske infrastrukture znaša 20.000 EUR,
- strošek začetnega razvoja informacijskega sistema bo znašal 70.000 EUR za razvoj osnovnih funkcionalnosti. Izračunan je na podlagi udejstvovanja oseb v projektni ekipi in cenika zunanjega izvajanja informatike,
- vsaka inkrementalna faza razvoja, ki je predvidena za obdobje 2 mesecev, je stroškovno ocenjena na 5.000 EUR. Pri tem se v začetku predvideva več iteracij (povprečno pet na leto) ter po prvem letu, ko bodo razvite vse glavne predvidene funkcionalnosti, povprečno 2-krat letno do četrtega leta, nato 1-krat letno,
- stroški prerazporeditve dela zaradi premestitve številnih nalog zaposlenih v organizaciji na druge zaposlene ter hkrati strošek zaposlenih, ki sodelujejo v ekipi razvoja, so znašali 15.000 EUR,
- strošek vzdrževanja rešitve znaša 300 EUR mesečno,

- letni prihranek iz naslova uravnavanja denarnih tokov znaša 70.000 EUR. Ta se je določil glede na pretekle podatke,
- prihranek z uporabo nove rešitve na področju valutnega tveganja znaša 130.000 EUR ter se vsako leto zmanjša za 10.000 EUR. Prihranek iz naslova obrestnega tveganja znaša 90.000 EUR na leto. Pri tem se ni upoštevalo ocene, da ima podjetje lahko pri izpostavljanju tveganjem tudi koristi,
- informacijski sistem je v prvem letu uporabe omogočil zmanjšanje dveh zaposlenih na področju finančne analize ter v drugem letu, ko so bile izvedene vse glavne funkcionalnosti sistema, dodatno še za enega zaposlenega.

Tabela 7: Projekcija vračanja investicije

		Leto 0	**Leto 1**	**Leto 2**	**Leto 3**	**Leto 4**	**Leto 5**
Strošek	Informacijska infrastruktura	20.000					
	Začetni razvoj informacijskega sistema	70.000					
	Inkrementalni razvoj		25.000	10.000	10.000	10.000	5.000
	Prerazporeditev dela	15.000					
	Vzdrževanje		3.600	3.600	3.600	3.600	3.600
	Kumulativni strošek skupno	90.000	108.600	122.600	135.800	149.400	158.000
Prihranek	Uravnavanje denarnih tokov		70.000	70.000	70.000	70.000	70.000
	Valutne pozicije			120.000	110.000	100.000	90.000
	Obrestne pozicije			90.000	90.000	90.000	90.000
	Stroški dela		60.000	80.000	80.000	80.000	80.000
	Kumulativni prihranek skupno		130.000	320.000	600.000	870.000	1.130.000
	Skupaj	–90.000	21.400	197.400	464.200	720.600	972.000
	Diskontni faktor	1	1,15	$(1,15)^2$ = 1,32	$(1,15)^3$ = 1,52	$(1,15)^4$=1,75	$(1,15)^5$ = 2,01
	Diskontiran denarni tok	–90.000	18.609	149.545	305.295	411.771	483.582

Iz Tabele 7 je razvidno, da je informacijski sistem že po prvem letu razvoja utemeljen. Metoda »doba vračanja investicije« ima v svoji osnovi pomanjkljivost, in sicer da ne upošteva časovne vrednosti denarja. Da se temu izognemo, lahko uporabimo diskontirano

dobo vračanja investicije, kjer se namesto nominalnih denarnih tokov uporabi diskontirane. Neto denarni tok v tabeli 7 je diskontiran z zahtevano stopnjo donosa lastnikov v višini 15 %.

Podjetja v večini primerov, ko delajo analizo stroškov in koristi uvedbe informacijske tehnologije v poslovne procese, uporabljajo metodo »neto sedanje vrednosti« (angl. *Net Present Value*, v nadaljevanju NSV). Predstavlja razliko med seštevkom vrednosti diskontiranih denarnih tokov (sedanja vrednost koristi) in sedanjo vrednostjo stroškov. Naložba je za podjetje sprejemljiva, če je NSV večja od 0 (če je NSV enaka 0, je organizacija do projekta indiferentna). Prednost te metode je upoštevanje vseh denarnih tokov in časovne vrednosti denarja (Laudon, 2013, str. 568–570).

Za naš primer izračunana vrednost NSV znaša 1.278.802 EUR, kar pomeni, da je naložba upravičena.

Metoda notranje stopnje donosa (angl. *Internal Rate of Return*, IRR) je obrestna mera, pri kateri je NSV enaka 0. Projekt je sprejemljiv, kadar je NSD višja od stroška kapitala danega podjetja (Laudon, 2009, str. 568–570).

NSD sem izračunal s pomočjo enačbe programa Excel in za naš primer za ekonomsko dobo petih let znaša **200 %**. Izračun pokaže, da je ta višja kot zahtevana stopnja donosnosti lastnikov (15 %) in je zato tudi upravičena.

7 KRITIČNA ANALIZA UPORABE IZBRANE METODOLOGIJE DSDM

Učinkovita uporaba metodologij razvoja informacijskih sistemov je vprašanje tako teoretičnega kot tudi praktičnega pristopa. Metodologija razvoja informacijskih sistemov je definirana kot organizirana zbirka konceptov, prepričanj, vrednot in normativnih načel podprtih z materialnimi viri (Ivari, 2001, str. 179–218).

V zadnjih desetletjih so se tradicionalne metode razvoja informacijskih sistemov izkazale kot vse preveč toge, kompleksne in neprimerne za učinkovit razvoj dinamičnih in modernih informacijskih sistemov, ki so zahteva moderno zasnovanih organizacij, katerih glavna lastnost je prilagajanje nenehnim spremembam. To v veliki meri zagotavljajo manj toge agilne metode razvoja, ki po svoji naravi niso tako strogo zasnovane. Tako ob premišljeni in dolgoročni uporabi dopuščajo prilagoditev pravil v obliko, ki je za organizacijo najprimernejša (Ivari, 2001, str. 179–218).

Kot sem povedal že v uvodu, je eden izmed ciljev magistrskega dela tudi pokazati najbolj problematične ter kritične vidike metodologij razvoja informacijskih sistemov in kako te

vidike prilagoditi v praksi tako, da uporabljena metodologija postane sprejemljivejša ter optimalnejša za uporabo v posamezni organizaciji.
Prilagoditve metod sledijo dvema osnovnima vidikoma (Baskerville, 2001, str. 11–27):

- **inženirski vidik** je osnovan na načelih znanosti in teorije in izhaja iz metodologije. Poudarja strukturne vidike metod in običajno uporablja znanstvenoteoretične koncepte za njihovo prilagoditev,
- **izkustveni vidik** deluje po principu, kako posamezna metoda deluje v okolju svoje dejanske uporabe in se odraža v socialno-organizacijskem znanju.

Dejstvo je, da je vsak projekt razvoja informacijskih sistemov unikaten ter so po drugi strani določene uporabljene razvojne metodologije preveč splošne. Tako je potrebna večja ali manjša prilagoditev glede na lastnosti projekta. Ker prilagajanje metod v organizaciji navadno poteka spontano, te niso formalno zapisane. Ravno iz tega razloga v magistrskem delu poleg opisa projekta razvoja informacijskega sistema podam tudi prilagoditve uporabljene formalno definirane razvojne metodologije DSDM.

Agilna metodologija DSDM je bila sprejeta kot privzeta metodologija za vse projekte razvoja informacijskih sistemov v organizaciji, kjer delam. Glavni razlog je bil doseg čim prej delujoče rešitve in kasneje njena postopna dodelava ter izpopolnjenje in uporaba enotnega izrazoslovja v vseh projektih. Metoda poudarja prav koncepte primernosti in prilagodljivosti do mere, ki je primerna za projekt ali organizacijo. Metodologija ne določa tehnik snovanja, ki so po navadi del orodij, ki jih uporabljamo, in ne metod. Ravno zaradi tega je uporaba DSDM-metodologije zelo prilagodljiva, saj jo lahko izvajamo v polni meri, uporabimo le posamezne tehnike iz njenega nabora ali samo njeno izrazoslovje. Metodologija vsebuje tudi t. i. »primernostni filter«, ki pomaga preveriti kritične dejavnike uspeha izbora DSDM in lastnosti projekta, kjer DSDM postane res učinkovit. Na njegovi osnovi se nato odločimo za uporabo DSDM-metodologije ali njeno ustrezno prilagoditev.

DSDM-izvedba v organizaciji je vodena prek projektnega vodja, ki ima bogate izkušnje iz vodenja številnih različnih projektov. Tako lahko pri vsakem projektu sprejme dobro odločitev o primernosti DSDM in določi stopnjo prilagoditve glede na zahteve ter lastnosti posameznega projekta. Če projektni vodja nima dovolj izkušenj, se na projekt kot trener doda posebnega bolj izkušenega projektnega vodji, ki nastopa kot svetovalec projektnemu vodji pri projektu. Za opredelitev projekta se uporablja razširjen in dopolnjen »primernostni filter« DSDM-metodologije z dodatnimi vprašanji, pojasnili, pripombami in nasveti, ki jih projektni vodje dobijo v času izvedbe projektov v organizaciji.

7.1 Kritični vidiki uporabe metodologije DSDM v organizaciji

V času izvajanja opisanega projekta smo ugotovili, da največjo skrb in probleme izvajanja DSDM-metodologije povzroča uresničevanje filozofije ter osnovnih tehnik metodologije.

Problematična se je izkazala predvsem popolnoma dosledna uporaba načel metodologije DSDM, ki so bolj podrobno opisana v poglavju 3.6. Naj navedem nekatere izmed njih.
Ugotovljeno je bilo, da je največkrat zelo težko zagotoviti načelo moči sprejetja odločitev razvojne skupine. Organizacijska struktura v našem okolju je postavljena hierarhično, informatika pa je le eno izmed področij organizacije, ki omogoča njeno delovanje, in ne ciljno področje delovanja organizacije. V sedanji organizacijski strukturi je zato težko doseči, da se udeležencem informatike da tako veliko moč, da lahko sprejemajo odločitve, ki so posredno pomembne tudi za organizacijo. Za vse odločitve se v našem okolju delovanja organizacije vedno zahteva odobritev višjih ravni znotraj organizacijske strukture, ki pa so zaradi narave svojega dela le malokrat lahko tudi neposredno udeleženi v razvojni ekipi.

Naslednji velik problem, ki smo ga zaznali v času izvedbe projekta po metodologiji DSDM, je problem treh načel: načela stalnega sodelovanja uporabnika, načela iterativnosti ter inkrementalnosti ter načela poslovnega namena. Problem se kaže predvsem v njihovem medsebojnem izključevanju. Metodologija predvideva več iteracij in inkrementalen razvoj, hkrati pa tudi aktivno sodelovanje uporabnika skozi celoten razvoj. To pomeni, da se morajo določene faze ter pripadajoče aktivnosti velikokrat ponavljati z vključitvijo stalne povratne informacije uporabnika. Tako mora biti uporabnik aktivno prisoten prek vseh iteracij. Tako velika prisotnost uporabnika ob razvoju pa odpre vprašanje poslovne upravičenosti tega in s tem tudi rešitve.

Prav tako je v projektu nemogoče ali zelo težko razporediti zahteve po načelu pravil MOSCOW ter jih pravilno razporediti v časovne okvire. Na začetku projekta imamo velikokrat premalo potrebnih informacij in jih pridobimo šele skozi projekt.

V naslednjem podpoglavju predstavim, kako zgoraj naštete probleme, ki smo jih zaznali v času izvedbe projekta ob uporabi DSDM-metodologije, rešimo s prilagajanjem metodologije glede na različno vrsto in zahteve projekta.

7.2 Prilagoditev kritičnih vidikov metodologije DSDM

Za vsakega izmed zgoraj zaznanih problemov smo poiskali ustrezno rešitev. Te so se naslanjale predvsem na izkustveni vidik prilagoditev uporabljene metodologije.

Za problem reševanja moči sprejetja odločitev razvojne skupine smo zaznali dve možni rešitvi. Pri prvi možnosti lahko eno izmed ključnih oseb v podjetju, ki ima moč odločanja, vključimo v razvojno skupino, vendar to v večini primerov ni mogoče. Poslovanje današnjega časa namreč zahteva osredotočenost ključnih oseb na opravljanje svojih vlog, ki se tako ne morejo v veliki meri posvetiti razvoju posamezne rešitve. Druga, bolj sprejemljiva možnost je bila, da je razvojna skupina sicer imela moč sprejemanja odločitev, vendar so morale biti vse sprejete odločitve nato pregledane in potrjene s strani

lastnika rešitve in vodstva. V praksi smo tako uporabili drugo možnost s še dodatno prilagoditvijo, da potrjevanje ni potekalo na ravni posameznih odločitev, ampak množice odločitev. Tako smo odločitve v razvojni skupini sprejemali bolj sistematično, obenem pa tudi razbremenili vodstvo pri njihovem potrjevanju.

Problem treh načel, načela stalnega sodelovanja uporabnika, načela iterativnosti ter inkrementalnosti ter načela poslovnega namena, smo rešili z nepopolno uporabo načela iterativnosti ter inkrementalnosti. Za inkrementalnost smo imeli na voljo dve izbiri: en inkrement (nobenih dodatnih pod rešitev) ali več inkrementov. Podobno izbiro smo imeli na voljo tudi glede iterativnosti: brez iteracij in več iteracij. Kombinacija možnosti nas je pripeljala do uporabe prilagojenega – hibridnega življenjskega razvojnega cikla metodologije DSDM. Glede na možnosti sodelovanja uporabnikov s strani lizing podjetja smo se odločili, da glavnino sistema, ki vsebuje ključne funkcionalnosti ter ogrodje sistema, razvijemo v enem inkrementu in več iteracijah. Za razvoj tega dela sistema je bil natančno določen časovni rok. Tako se je vodstvo lizing podjetja lahko zavedalo obdobja, v katerem je mogoča motnja zaposlenih, ki nastopajo v vlogi uporabnikov sistema ter tako sodelujejo pri projektu. Obenem smo lahko določili funkcionalnosti, ki so ključne, in jim dali najvišjo prioriteto. Za vse funkcionalnosti po tem obdobju pa se je sprejela odločitev, da se izvajajo inkrementalno brez iteracij. Omejila se je tudi dolžina posameznega inkrementa na obdobje dveh mesecev. To se je določilo na osnovi predvidenih denarnih sredstev in človeških virov. Vodstvo je tako natančno opredelilo redni strošek, ki ga bo vsak cikel nadaljnjega razvoja zahteval, obenem pa kolikor se je dalo omejilo motnjo zaposlenih samo na zaposlene, ki so bili udeleženi pri razvoju trenutne funkcionalnosti inkrementa. Ker je obdobje inkrementa dovolj kratko, dodatne iteracije niso potrebne, motnja zaposlenih, ki so nastopali kot uporabniki, pa je tako enkratna. Zaradi izločitve dodatnih iteracij pa se je povečalo tveganje za odstopanje od dejanskih zahtev in želja uporabnikov. To tveganje je bilo sprejeto s strani vodstva. Rešitev, ki je bila predvidena v primerih, kadar je prišlo do odstopanj ali neskladnosti inkrementa z zahtevami, je, da se vsa neskladja odpravljajo v naslednjih inkrementih kot nova funkcionalnost rešitve.

Rešitev gornjega problema je rešila tudi problem razporejanja zahtev po načelu MOSCOW in časovno uokvirjanje teh zahtev. Ker je bil rok izvedbe glavnine funkcionalnosti znan, prav tako pa tudi ključna funkcionalnost, smo za ta del lahko klasično uporabili načela MOSCOW ter tako opredeljene zahtevke razporedili v časovne okvire, kot predvideva metodologija DSDM. Prav tako smo pri nadaljnjih inkrementih, ki so bili omejeni na časovno obdobje dveh mesecev, uporabljali načela MOSCOW. Pri tem smo upoštevali pravilo, da vse želene funkcionalnosti, ki niso prišle v trenutni inkrement, vodimo na skupnem seznamu in jih imamo v obravnavi v naslednjih inkrementih razvoja.

Za prilagoditve metodologije bi velikokrat težko našli teoretično ozadje – v večini predstavljajo zmes izkušenj in znanja projektnih vodij. Ti za iskanje rešitev vseh kritičnih vidikov metodologije v okviru posameznega projekta proučujejo tako njeno filozofijo,

okvir kot tudi njene bistvene tehnike. Samo tako lahko posamezno metodologijo standardiziramo za uporabo pri vseh projektih na ravni organizacije in tako v veliki meri zagotovimo tudi prenos obstoječega znanja med projekti.

SKLEP

Finančna podjetja se pri opravljanju svoje dejavnosti soočajo z najrazličnejšimi tveganji. Prevzemanje tveganja finančnega podjetja je ena izmed funkcij, s katero se opravičuje njegov obstoj. Če želijo uspešno in učinkovito opravljati svojo temeljno funkcijo finančnega posredništva, morajo znati upravljati s tveganji. Tveganje morajo znati prepoznati, pravilno izmeriti in obvladovati. Pri tem jim je lahko v veliko pomoč ustrezen vpogled v podatke v taki obliki, ki omogoča obvladovanje teh tveganj. Za dosego tega cilja je bilo v opazovani finančni instituciji treba razviti informacijski sistem, ki omogoča vpogled v finančne podatke institucije, na osnovi katerih se finančni analitiki institucije lahko pravilno odločajo in pripravijo ustrezno strategijo obvladovanja tveganj.

V procesu razvoja informacijskih sistemov se postavlja temeljno vprašanje, ali bo informacijski sistem izdelan v roku, s čim manj napakami in pomanjkljivostmi ter s čim bolj zadovoljnim naročnikom ter kupcem. Kot odgovor na to vprašanje so bile izdelane številne metodologije razvoja informacijskih sistemov. Ti se neprestano razvijajo in izboljšujejo ter tako ponujajo vse novejše in učinkovitejše pristope, ki odgovarjajo na vseskozi prisotne probleme razvoja informacijskih sistemov.

Izbira najustreznejšega pristopa je ključna za uspeh projekta kot tudi za vodenje projekta najboljših praks. Pri tem imamo možnost izbire od začetnih klasičnih pa vse do novejših agilnih pristopov razvoja informacijskih sistemov, ki so manj obsežni, a hkrati manj procesno usmerjeni ter tako bolj osredotočeni na končni rezultat – rešitev.

Poglavitni cilj magistrskega dela je bila analiza informacijskih potreb in izdelava delujoče rešitve za spremljanje likvidnostnega tveganja ter njena uvedba v podjetje. Magistrsko delo tako vsebuje predstavitev razvoja informacijskega sistema za upravljanje z likvidnostnim tveganjem v finančni instituciji skozi njen celoten življenjski cikel.

Ta se prične na začetku, kjer sprejmemo odločitev za razvoj, in se nadaljuje prek odločitev o izbiri ustreznih pristopov razvoja ter ustrezne razvojne metodologije. Ta sama po sebi ni univerzalna rešitev oziroma rešitev vseh problemov projekta razvoja informacijskega sistema. V veliki meri je odvisna od številnih dejavnikov, ki nastopajo znotraj projekta razvoja informacijskega sistema ter njegovega okolja. Tako je treba vsako še tako premišljeno izbrano metodologijo ustrezno prilagoditi glede na lastnosti, zahteve in okolje posameznega projekta razvoja. Šele takrat razvojno metodologijo v okviru projekta razvoja naredimo res uporabno. Pri tem lahko izkoristimo možnosti prilagoditev, ki jih predvideva

metodologija kot tudi vse izkustvene prilagoditve, ki so v veliki meri odvisne od znanj projektnih vodij in tudi ostalih udeležencev v projektu.
Vsak razvoj novega informacijskega sistema zahteva tudi analizo ter obenem optimizacijo vseh obstoječih procesov, ki jih bo nova rešitev vključevala. V prikazanem primeru razvoja je bil glaven proces spremljanje likvidnosti v lizing podjetju. Podrobna analiza obstoječega stanja tega procesa je bila ključna za ves nadaljnji razvoj. Analiza stanja je bila še toliko pomembnejša, ker v lizing podjetju do sedaj proces ni bil formalno določen. Temeljil je na izvajanju postopkov nekaj zaposlenih oseb. Analiza in optimizacija tega sta tako v veliki meri pripomogli tudi k formalni ureditvi izvajanja procesa spremljanja likvidnosti.

Že na začetku projekta razvoja je bila želja vodstva lizing podjetja, da se najprej razvije ključen del procesa podpore spremljanja likvidnosti. Po uvedbi tega in glede na njeno uspešnost pa se bo razvilo tudi ostale funkcionalnosti. Zasnova informacijskega sistema je tako morala biti izdelana dovolj premišljeno, da je dovoljevala možnost kasnejšega vključevanja novih funkcionalnosti in modulov v sistem. K temu me je v veliki meri vodila že izbrana metodologija razvoja informacijskih sistemov, ki v svoji osnovi vsebuje inkrementalen pristop gradnje informacijskih sistemov.

S predstavitvijo razvoja informacijskega sistema za upravljanje z likvidnostnim tveganjem v finančni instituciji skozi njen celoten življenjski cikel sem hkrati želel predstaviti tudi osnove razvoja informacijskih sistemov, ključne faze razvoja v okviru izbrane metodologije ter rešitev predstaviti kot predlogo za reševanje podobne problematike v finančnih institucijah ali organizaciji s podobnimi zahtevami.

Vsak projekt razvoja informacijskih sistemov je unikaten, uporabljene razvojne metodologije pa preveč splošne, da bi optimalno podajale rešitve za vse probleme razvoja za posamezno organizacijo in projekt. Sprejeta metodologija je tako ogrodje pristopov razvoja rešitve. Za njeno optimalno izvajanje je temu treba dodati večje ali manjše prilagoditve glede na lastnosti projekta in organizacije. S pričujočim delom sem želel izpostaviti tudi to problematiko in predstaviti njeno reševanje pri opisanem projektu.

Razvoj informacijskega sistema se zaključi, ko rešitev vpeljemo v produkcijsko okolje in je tako predana v uporabo. Z dosego tega stanja je bil glavni cilj magistrskega dela dosežen. Življenjski cikel razvitega informacijskega sistema pa se v tem trenutku ne zaključi. Nadaljuje se v obliki nadaljnjega nadgrajevanja ter razvoja kot tudi vzdrževanja razvitega informacijskega sistema.

Informacijski sistem je živ, treba ga je vseskozi razvijati, dograjevati ter prilagajati novi tehnologiji in novim standardom na področju procesa, ki ga pokriva. V okviru predstavljene rešitve se to brez večjih problemov redno izvaja že pet let. Tudi to je velik pokazatelj, da je bil sistem dobro zasnovan in da je bila sprejeta metodologija dela prava izbira.

LITERATURA IN VIRI

1. Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). *Agile software development methods.* Najdeno 4. novembra 2011 na spletnem naslovu http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf
2. Allen, S. (2003). *Financial Risk Management: A Practitioner's Guide to Managing Market and Credit Risk*. NY: John Wiley & Sons, Inc.
3. Armour, P. G. (2004). *The Laws of Software Process: A New Model for the Production and Management of Software*. Broken Sound Parkway: Auerbach Publications.
4. Attarzadeh, I., & Hock, O. S. (2008). *Project Management Practices: The Criteria for Success or Failure.* Najdeno 20. oktobra 2011 na spletnem naslovu http://www.ibimapublishing.com/journals/CIBIMA/volume1/v1n28.pdf
5. Avison, E. D., & Fitzgerald, G. (2003). *Information Systems Development: Methodologies, Techniques and Tools* (3rd ed.). London: McGraw-Hill.
6. Baskerville, R., & Stage, J. (2001). Accommodating Emergent Work Practices: Ethnographic Choice of Method Fragments. In B. Fitzgerald, N. Russo & J. I. DeGross (Eds.), *In realigning research and practice: The social and organizational perspectives*. Boston: Kluwer Academic Publishers.
7. Beck, K. (2005). *Extreme Programming Explained: Embrace Change*. New Jersey: Pearson Education, Inc.
8. Beck, K., & Fowler, M. (2000). *Planning Extreme Programming*. Boston: Addison Wesley Professional.
9. Berk, A., Peterlin, J., & Ribarič, P. (2005). *Obvladovanje tveganja: Skrivnosti celovitega pristopa*. Ljubljana: GV Založba.
10. Bessis, J. (2002). *Risk Management in Banking*. Wese Sussex: John Wiley & Sons
11. Bobek, S. (2007). *Strateški informacijski sistemi*. Najdeno 10. oktobra 2011 na spletnem naslovu http://epf-oi.uni-mb.si:8000/clani/bobek/FI/Strateski_is.pdf
12. Boehm, B. W. (1988). *A Spiral Model of Software Development and Enhancement*. Najdeno 29. novembra 2011 na spletnem naslovu http://www.cs.umd.edu/class/ spring2003/cmsc838p/Process/spiral.pdf
13. Brigham, E. F., & Ehrhardt, M. C. (2004). *Financial management*. Ohio: South-Western.
14. CA Technologies (2010): *Balancing agility with governance*. Najdeno 29. oktobra 2011 na spletnem naslovu http://www.ca.com/~/media/files/supportingpieces/ ppm-summit-agile-research-oct-2010.aspx
15. Cappelan, C., Chu, J., Patel, K., & Yang, Z. (2004). *Spiral Software Development, Computer Science 180 - Software Engineering.* Medford, Massachusetts: Tufts University.
16. Cockburn, A. (2001). *Agile Software Development*. Boston: Addison Wesley Professional.

17. Čibej, J. A. (2006). Investicije. *E-revir*. Najdeno 2. aprila 2012 na spletnem naslovu http://www.erevir.si/Moduli/Clanki/JAC_ppo/JAC_EREVIR_060516_Investicije.pdf
18. *Investicije – ocenjevanje investicij DSDM Public Version 4.2 Manual*. Najdeno 4. novembra 2011 na spletnem naslovu http://intra.iam.hva.nl/content/0708/propedeuse/ontwikkelmethoden_en_technieken//intro-en-materiaal/DSDM.pdf
19. Fitzgerald, B. (1996). *Formalised Systems Development Methodologies: A Critical Perspective*. Najdeno 25. oktobra 2011 na spletnem naslovu http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.105.3317&rep=rep1&type=pdf
20. Fowler, M. (2003). *The New Methodology Thoughtworks*. Najdeno 4. novembra 2011 na spletnem naslovu http://www.martinfowler.com/articles/newMethodology.html
21. West, D., & Grant, T. (2010). *Agile Development: Mainstream Adoption Has Changed Agility*. Najdeno 29. oktobra 2011 na spletnem naslovu http://www.osp.ru/netcat_files/18/10/h_d8eddd303b6cf0c38c23601c4363bee4
22. Gams, M. (1998). *Informacijska družba*. Ljubljana: DZS.
23. Gradišar, M., Jaklič, J., & Turk, T. (2007). *Osnove poslovne informatike*. Ljubljana: Ekonomska fakulteta.
24. Gradišar, M., & Resinovič, G.(2001). *Informatika v poslovnem okolju*. Ljubljana: Ekonomska fakulteta.
25. Hajne, S. (2011). *Razvoj informacijskega sistema za upravljanje z nekreditnimi tveganji v finančnem podjetju* (magistrsko delo). Ljubljana: Ekonomska fakulteta
26. *ICommerce Design Strategies*. Najdeno 12. decembra 2014 na spletnem naslovu http://www.theserverside.com/news/1364722/ICommerce-Design-Strategies
27. Iivari, J., Hirschheim, R., & Klein, H.K. (2001). A dynamic framework for classifying information systems development methodologies and approaches. *J. of Management Information Systems*, *17*(3), 179–218.
28. Kappe, B. (2009). *The Five Deadly Sins Of Software Development*. Najdeno 20. oktobra 2011 na spletnem naslovu http://pathfindersoftware.com/2009/04/the-five-deadly-sins-of-project-management/
29. Jenkins, N. (2008). *A Software Testing Primer*. Najdeno 20. januarja 2015 na spletnem naslovu http://www.nickjenkins.net/prose/testingPrimer.pdf
30. Ključevšek, B. (2001). *Prenova informacijskega sistema s standardnimi rešitvami* (diplomsko delo). Ljubljana: Ekonomska fakulteta.
31. Kneiberg, H. (2007). *»Scrum And XP From The Trenches«*. New York: InfoQ.com
32. Kovačič, A. (1999). Najboljše programske rešitve ter pravi izvajalci? *Uporabna informatika*, 2(VII), 39–42.
33. Laudon, K. C., & Laudon, J. P. (2013). *Management Information Systems - Managing the Digital Firm*. Essex: Pearson Education Limited
34. Leffingwell, D., & Widrig, D. (1999). *Managing Software Requirements*. Boston: Addison Wesley.
35. Lesjak, D. (2003). *Pravna informatika – uvodna poglavja*. Najdeno 10. oktobra 2013 na spletnem naslovu http://www.pf.uni-mb.si/pravna_informatika/images/pi_1.pdf

36. *Manifesto for Agile Software Development*. Najdeno 10. septembra 2011 na spletnem naslovu http://www.agilemanifesto.org
37. Marchesi, M., Succi, G., Wells, D., & Williams, L. (2002). *Extreme Programming Perspectives*. Michigan: Addison-Wesley Pub Co.
38. Matz, L., & Neu, P. (2007). *Liquidity Risk Management*. Chichester: John Wiley & Sons.
39. McConnel, S. (1996). *Rapid Development, Microsoft Press.* Redmond: A Division of Microsoft Corporation.
40. *Microsoft .NET*. Najdeno 10. septembra 2013 na spletnem naslovu http://www.microsoft.com/net
41. Miller, G. G. (2001). *The Characteristics of Agile Software Processes.* Najdeno 23. oktobra 2010 na spletnem naslovu http://faculty.salisbury.edu/~xswang/Research/Papers/SERelated/Agile/12510385.pdf
42. Mnkandla, E. (2008). *A selection framework for agile methodology practices*. Johannesburg: *Založba*.
43. PM Solutions. (2005). *Selecting a Software Development Life Cycle (SDLC).* Methodology, Project Management Solutions, Inc. & Park Hill Technologies LLC. Project Management Solutions, Inc. & Park Hill Technologies LLC.
44. Prohaska, Z. (2004). *Finančni trgi*. Ljubljana: Ekonomska fakulteta.
45. Racconn, L. B. S. (1995). The Chaos Model and the Chaos Life Cycle, ACM Software Engineering Notes. *ACM Press, 20*(1), 55–66.
46. Saunders, A. (2000). *Financial institutions management: a modern perspective.* Boston: Irwin/McGraw-Hill.
47. Shore, J., & Warden, S. (2007). *The art of agile Development*. Sebastopol: O'Reilly.
48. Srića, V., Treven, S., & Pavlić, M. (1995). *Informacijski sistemi*. Ljubljana: Gospodarski vestnik.
49. Silič, M., & Krisper, M. (2003). *Enotna metodologija razvoja IS*. Ljubljana: Center Vlade RS za informatiko.
50. Sinkey, F. J. (1998). *Commercial Bank Financial Management*. New Jersey: Prentice Hall.
51. *Kdo smo – Skupina podjetij Globus Marine International*. Najdeno 10. oktobra 2011 na spletnem naslovu http://www.g-gmi.si/gmiweb/Documents/Kdo_smo_5.aspx
52. Solina, F. (1997). *Projektno vodenje razvoja programske opreme*. Ljubljana: Fakulteta za računalništvo in informatiko.
53. Stair, R., & Reynolds, G. (2010). Principles of Information Systems, Course Technology Cengage Learning. Boston: Cengage Learning.
54. Stoimen, P. (2011), *Main Features of the Traditional Software Development Methodologies.* Najdeno 26. oktobra 2014 na spletnem naslovu http://www.stoimen.com/blog/2011/04/19/main-features-of-the-traditional-software-development-methodologies/
55. *Top-down and bottom-up design*. Najdeno 10. septembra 2011 na spletnem naslovu http://en.wikipedia.org/wiki/Top-Down_Model

56. Turk, I. (1987). *Pojmovnik poslovne informatike*. Ljubljana: Društvo ekonomistov Ljubljana.
57. The Standish Group International (2009). *CHAOS Sumary 2009*. Najdeno 20. oktobra 2013 na spletnem naslovu http://standishgroup.com/newsroom/chaos_2009.php
58. The Standish Group International (2009). *CHAOS Rising: A CHAOS Executive Commentary*. Najdeno 20. oktobra 2013 na spletni strani http://standishgroup.com/index.php
59. Ward, J., & Peppard, J. (2002). *Strategic Planning for Information System*. West Sussex: John Wiley & Sons.
60. Wysocki, R. K., & McGary, R. (2003). *Effective Project Management: Traditional, Adaptive, Extreme* (3rd Ed.). Indianapolis: Wiley Publishing, Inc..