UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

SPECIALISTIČNO DELO

MANAGEMENT TVEGANJ PRI PROJEKTIH RAZVOJA VGRAJENE PROGRAMSKE OPREME

Ljubljana, april 2016 MARJAN HANČ

IZJAVA O AVTORSTVU

Spodaj podpisani Marjan Hanč, študent Ekonomske fakultete Univerze v Ljubljani, izjavljam, da sem avtor specialističnega dela z naslovom Management tveganj pri projektih razvoja vgrajene programske opreme, pripravljenega v sodelovanju s svetovalcem, docentom dr. Aljažem Staretom.

Izrecno izjavljam, da v skladu z določili Zakona o avtorskih in sorodnih pravicah (ur. l. RS., št. 21/1995, s spremembami), dovolim objavo specialističnega dela na fakultetnih spletnih straneh.

S svojim podpisom zagotavljam, da

- je predloženo besedilo rezultat izključno mojega lastnega raziskovalnega dela;
- je predloženo besedilo jezikovno korektno in tehnično pripravljeno v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani, kar pomeni, da sem

    - poskrbel, da so dela in mnenja drugih avtorjev oz. avtoric, ki jih uporabljam v specialističnem delu, citirana oz. navedena v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani, in
    - pridobil vsa dovoljenja za uporabo avtorskih del, ki so v celoti (v pisni ali grafični obliki) uporabljena v tekstu in sem to v besedilu tudi jasno zapisal;

- se zavedam, da je plagiatorstvo – predstavljanje tujih del (v pisni ali grafični obliki) kot mojih lastnih – kaznivo po Kazenskem zakoniku (Ur. l. RS 55/2008, s spremembami);
- se zavedam posledic, ki bi jih na osnovi predloženega specialističnega dela, dokazano plagiatorstvo lahko predstavljalo za moj status na Ekonomski fakulteti Univerze v Ljubljani v skladu z relevantnim pravilnikom.

V Ljubljani, dne 21. 4. 2016 Marjan Hanč

KAZALO

KAZALO SLIK

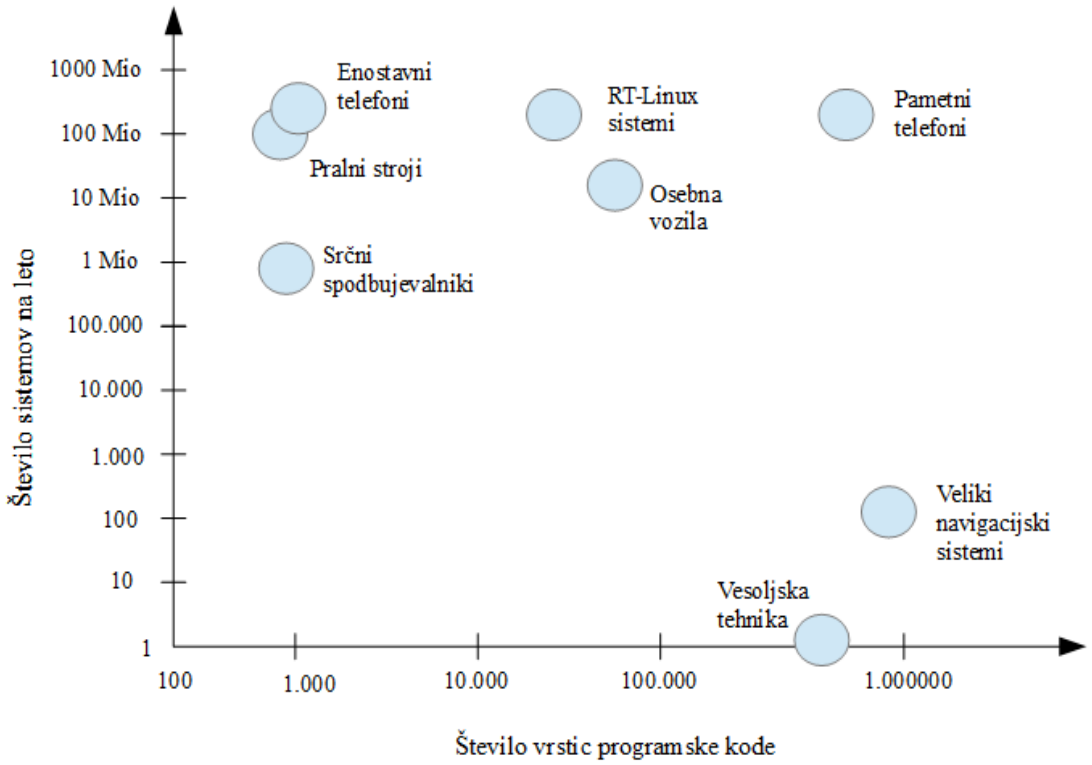
KAZALO TABEL

UVOD

Projektni management na razvojnih projektih programske opreme, še posebej vgrajene programske opreme (angl. *Embedded Software*), na kateri bo poudarek v tem specialističnem delu, je zelo interdisciplinaren in obenem zelo tehničen.

Razlog za to leži v samem namenu vgrajene programske opreme, saj z njo upravljamo strojno opremo (elektronsko-mehanski sklop), ki je lahko npr. enostavni krmilnik centralne kurjave, sodobni merilnik električnega toka z daljinskim javljanjem podatkov, pametni telefon ali visoko kompleksna krmilno procesna enota v sodobnem avtomobilu, ki voznika opozarja na nevarnosti ali celo samostojno upravlja z vozilom. Kompleksnost programske kode glede na področje uporabe in ocenjeno velikost trga posameznih naprav prikazuje Slika 1.

Slika 1: Kompleksnost posameznih vrst sistemov glede na obseg programske kode



Vir: C. Ebert, & C. Jones, Embedded Software: Facts, Figures and Future, 2009, str. 43.

Vgrajena programska oprema doživlja s porastom informatizacije in avtomatizacije vsakdanjega življenja ekstremno tržno rast.

Po tržni analizi Transparency Market Research-a (2013) je bila vrednost trga vgrajene programske opreme v letu 2011 ocenjena na 121 milijard USD, s pričakovano povprečno letno rastjo 6,8 % v letih 2012–2018. Po najnovejših ocenah (Grand View Research, 2014) pa naj bi v letu 2013 že dosegala vrednost 140,32 milijard USD. Približno enaka razmerja veljajo

tudi v Evropi. Po analizah BITKOM-a (2010) je bila velikost trga vgrajene programske opreme v Nemčiji v letu 2010 ocenjena na 19 milijard EUR, z ocenjeno letno stopnjo rasti do 8,00 %.

Takšno rast trga na podlagi omenjenih analiz pospešuje predvsem povpraševanje po pametnih napravah, kot so npr. mobilne komunikacijske naprave in daljinski merilniki energije. Uporaba večjedrnih procesorjev z nizko porabo energije pospešuje razvoj novih pametnih naprav, hkrati pa naraščajoča ponudba na globalnem trgu znižuje ceno komponent. Vsi ti dejavniki pospešujejo množično uporabo v vsakdanjem življenju, s tem pa pojav novih podjetij, ki se ukvarjajo s tem poslovnim področjem.

Praksa kaže, da se večina razvojnih projektov vgrajene programske opreme srečuje s kroničnim preseganjem predvidenih stroškov in časovne izvedbe. Po navedbah McManus-a (2004, str. 2) je samo dobrih 10 % projektov razvoja programske opreme zaključenih v predvidenem roku in v okviru predvidenih stroškov, pa še ti so večinoma funkcijsko pomanjkljivi in kvalitativno slabši kot bi pričakovali.

McManus nadalje navaja rezultate raziskave Standish Group-a iz leta 1995, v kateri je bilo, na podlagi analize podatkov o poteku projektov, ugotovljeno naslednje:

- 16 % projektov je bilo učinkovito izvedenih, torej v planiranem roku in finančnem okviru (uspešni projekti);
- 31 % projektov je bilo predčasno ustavljenih in nedokončanih (neuspešni projekti);
- 53 % projektov je bilo zaključenih, ampak je bila cena izvedbe višja od predvidene, kar je imelo za posledico okrnitev prvotno zahtevanega funkcijskega obsega (težavni projekti);
- povprečna prekoračitev stroškov zaključenih ali predčasno ustavljenih projektov je za 190 % presegalo predvidene stroške izvedbe;
- v povprečju so zaključeni in ustavljeni projekti presegli prvotno načrtovani čas izvedbe za 222 %;
- zaključeni projekti so v povprečju izpolnjevali le 31 % prvotno zahtevanega funkcijskega obsega.

Žal se razmere vse od leta 1995 do danes, kljub intenzivnemu razvoju prakse managementa projektov, niso bistveno spremenile.

Tako je bil leta 2009, po raziskavah Standish Group-a (2009, str. 1–4), delež uspešno izvedenih projektov le 32 %, medtem ko je bil delež težavnih projektov 44 % in delež neuspelih projektov 24 %. Tudi najnovejši podatki iste raziskave Standish Group-a (2013, str. 1–5) iz leta 2013 kažejo, da je delež uspešnih projektov še vedno le 39 %, delež težavnih projektov 43 % in delež neuspelih projektov 18 %. Če upoštevamo podatek, da je v letu 2013 največji tržni delež vgrajene programske opreme od skupne vrednosti 140,32 milijard USD zavzemal segment avtomobilske elektronike (20,8 %) oz. 29,18 milijard USD (Grand View Research, 2014), lahko sklepamo, da so izgube zaradi slabo ali nepravočasno izvedenih projektov ogromne.

Po raziskavah Standish Group-a iz leta 2013 je bil izvedbeni čas „težavnih projektov“ v povprečju prekoračen za 74 %. Preračunano npr. na segment avtomobilske elektronike, to pomeni, da so bili, če so predstavljale investicije v letu 2013 le 20 % vrednosti, razvojni projekti v avtomobilskem segmentu dražji za dobro 1 milijardo USD. Čeprav se stanje izboljšuje in je področje managementa tveganj v tuji literaturi zelo dobro obdelano, se še vedno zastavlja vprašanje, kako je mogoče, da je delež uspešnih projektov po skoraj 20 letih intenzivnega razvoja stroke še vedno relativno nizek.

Če je pri klasičnih projektih mogoče razdeliti projektno nalogo na določena opravila in jih oceniti, je to pri projektih razvoja programske opreme praktično nemogoče. Pri klasičnem projektu, npr. pri gradnji hiše, lahko na osnovi izvedbenega projekta dokaj dobro ocenimo potrebno količino gradbenega materiala, potrebnega dela in časa za dokončanje hiše.

Pri razvoju programske opreme takšne ocene ne moremo podati. Za merjenje produktivnosti bi sicer lahko uporabili orientacijsko vrednost, ki pravi, da pri povprečnem razvojnem projektu izdelamo 200–350 vrstic[1] uporabne, ne-komentirane kode v enem inženirskem mesecu (Grady, 1992, str. 15–16; Jones, 2012, str. 6). Kljub temu ne moremo vnaprej predvideti, koliko vrstic programske kode bo potrebnih za izvedbo določene projektne naloge. Prav tako ne moremo predpisati ali zahtevati, da izvajalec v določenem času napiše določeno število vrstic programske kode. Iz tega sklepamo, da se pri načrtovanju projektov lahko zanašamo zgolj na izkušnje iz prejšnjih projektov, da čim bolje identificiramo tveganja (Boehm, 1989, str. 57) in na podlagi le-teh planiramo potrebne varnostne rezerve.

S premišljenim managementom tveganj lahko zmanjšamo verjetnost negativnega izida projekta, saj vnaprej poskušamo predvideti čim več tveganj in se pripravimo na njihovo obvladovanje. Vseh možnih tveganj vnaprej ne moremo predvideti, saj bi to zahtevalo preveč časa in energije, zato je obvladovanje tveganj na projektih dinamičen proces, ki mu moramo nujno nameniti dovolj pozornosti in ga podpreti z ustrezno komunikacijo znotraj in zunaj projekta. Proces obvladovanja tveganj na projektu ne vključuje samo obvladovanje negativnih tveganj (težav), ampak tudi odziv na pozitivna tveganja (priložnosti), ki lahko v določenih trenutkih pomagajo zasukati projekt bolje, kot je bilo prvotno predvideno.

Če povzamemo uvodne ugotovitve, se kaže pomen planiranja managementa tveganj (Kerzner, 2001, str. 156) tudi pri razvojnih projektih vgrajene programske opreme v dveh dejavnikih, in sicer v: tehnični nezmožnosti izvedbe in/ali slabem managementu tveganj. Kerzner (2001, str. 156) pravi, da ni jasne ločnice med tem, ali je za slab rezultat projekta kriva zgolj tehnična nezmožnost ali slab management, ampak je v večini primerov mešanica obojega.

Namen naloge je, da managerjem projektov s praktičnimi napotki pomagamo z bistvenimi elementi identifikacije, ocene in obvladovanja tveganj, ki se jih da v praksi, pri projektih razvoja vgrajene programske opreme učinkovito uporabiti in tako prihraniti na času in energiji. S tem želimo prispevati k učinkovitejši izvedbi projektov v proučevanem podjetju in izven, hkrati pa prispevati k razvoju stroke managementa tveganj in projektnega managementa nasploh, kot tudi spodbuditi znanstveno razpravo na tem področju.

[1] Pri tem mislimo na programiranje v visokih programskih jezikih npr. C, C++, Java ipd.

Cilj specialističnega dela je, iz splošne teorije managementa tveganj in v literaturi opisanih splošnih metod in spoznanj, ki jih različni avtorji opisujejo na primerih iz prakse, izluščiti bistvene elemente managementa tveganj, ki jih lahko managerji projektov razvoja vgrajene programske opreme učinkovito uporabijo v praksi.

Metodologija, ki jo bomo uporabili: s pomočjo proučevanja znanstvene literature, strokovnih člankov in virov na spletu bomo poskušali identificirati bistvene elemente managementa tveganj, ki jih lahko učinkovito uporabimo pri projektih razvoja vgrajene programske opreme.

V specialističnem delu bomo najprej na kratko podali splošen opis vgrajene programske opreme, specifične lastnosti in posebnosti teh projektov. V nadaljevanju bomo opisali vire tveganj, od zgoraj navzdol (angl. *Top to the bottom*), in sicer tista tveganja, ki izhajajo iz organizacije same, vrste projektov, uporabljenih razvojnih modelov programske opreme in projektov samih. Pri virih tveganj bomo poskušali, na podlagi strokovnih člankov in literature, zbrati čim več tveganj, ki so relevantna za obravnavano področje in jih sistematično razporediti.

Pri opisu procesa managementa tveganj bomo iz priporočil strokovne literature izbrali najprimernejši proces in obdelali posebnosti, ki se nanašajo na projekte razvoja vgrajene programske opreme.

S pomočjo teoretičnega dela bomo analizirali projekt X, ki se je odvijal v štirih fazah. Na proučevanem projektu bomo pokazali, zakaj je prihajalo v začetnih fazah, zaradi pomanjkljivega managementa tveganj, do težav in kako smo se z upoštevanjem metod in procesov, opisanih v teoretičnem delu, slednjim v sklepni fazi projekta izognili.

1 TVEGANJA PRI RAZVOJU VGRAJENE PROGRAMSKE OPREME

1.1 Značilnosti vgrajene programske opreme

Na splošno lahko vgrajeno programsko opremo in razvojne projekte primerjamo s standardno informacijsko tehnološko (v nadaljevanju IT) oz. aplikacijsko programsko opremo. Pri tem z IT oz. aplikacijsko programsko opremo označujemo tisto programsko opremo, s katero se srečujemo pri vsakodnevnem delu z računalnikom (npr. Word, Excel, Outlook itd.). Vgrajena programska oprema nam kot uporabnikom med delovanjem ni vidna, razen kakšne utripajoče lučke na armaturni plošči v avtomobilu ali prikaza temperature na sobnem termostatu, vendar zaradi tega ni nič manj kompleksna (npr. vodenje rakete).

Posebnost projektov vgrajene programske opreme je, da je projektna naloga le v redkih primerih sestavljena iz izdelave le ene komponente oz. aplikacije. Pri tovrstnih projektih se tako v večini primerov srečujemo z večnivojskim aplikativnim razvojem. Na najnižjem nivoju gre za izdelavo programske opreme, ki deluje neposredno s strojno opremo, tako da le-to nadzoruje in z njo izmenjuje podatke o merilnih in krmilnih vrednostih. V novejšem času imamo na tem nivoju opravka s programiranjem celotnega sistema na integriranem vezju

(angl. *System on the chip*). Klasičnemu razvoju (IT) programske opreme se približamo šele na višjih nivojih, ko izmenjujemo informacije z uporabnikom naprave in obdelujemo uporabniške podatke in zahteve. Bistvena razlika je, da imamo pri IT razvojnih projektih v osnovi že delujoč sistem, pri vgrajeni programski opremi pa z našo programsko opremo sistem šele spravimo v delovanje.

Za razliko od IT aplikacijske programske opreme izhaja večina funkcijskih zahtev, ki jih moramo izpolniti pri izvedbi projekta, iz vgrajene strojne opreme (angl. *Embedded hardware)* in le manjši del iz interakcije z uporabnikom. Posledično morajo biti izvedbene zahteve (angl. *Requirements*) zelo natančno opredeljene že pred samim začetkom projekta, saj lahko vsaka najmanjša sprememba povzroči t. i. inflacijo sprememb (angl. *Requirements inflation*) na vseh nivojih. Npr. majhna sprememba na nivoju strojne opreme lahko sproži veliko sprememb na vseh nivojih aplikativnega dela programske opreme.

Vgrajena programska oprema se večino časa izvaja sinhrono in v realnem času (angl. *Time triggered*), medtem ko se klasična programska oprema izvaja odzivno (angl. *Event triggered*). Npr. šele takrat, ko uporabnik pritisne tipko ali klikne z miško na določeno področje. Zaradi tega je odpravljanje napak v vgrajeni programski opremi zelo zahtevno in zahteva ogromno časa, saj napak(e) v večini primerov ni enostavno reproducirati.

Če razvoj programske opreme poenostavimo zgolj na posamezni programski sloj (angl. *Software application layer*) ali vgrajeno programsko komponento (angl. *Embedded software component*), lahko takšen projekt primerjamo s standardnim IT oz. aplikacijskim razvojnim projektom, saj zanj veljajo enake zakonitosti. V nadaljevanju tako ne bomo delali razlik med obema vrstama projektov, ampak bomo v primerih, ko je razlike vseeno potrebno upoštevati, na to posebej opozorili.

1.2 Tveganja povezana z ravnjo organizacijske zrelosti

Že v 90-ih letih, v začetni fazi masovne uporabe računalnikov in s tem tudi poplave namenske in nenamenske programske opreme, je postalo jasno, da obstajajo velike razlike med dobavitelji, tj. med podjetji, ki razvijajo oz. ponujajo programsko opremo na trgu ali jo razvijajo po naročilu. Watts Humprey je leta 1989 (Humprey, 1989) v okviru Software Engineering Institut-a na univerzi Carnegie Mellon postavil t. i. Capability Maturity Model (v nadaljevanju CMM) za objektivno ocenjevanje ravni organizacijske zrelosti podjetij, ki razvijajo in dobavljajo programsko opremo ameriškim vladnim organizacijam.

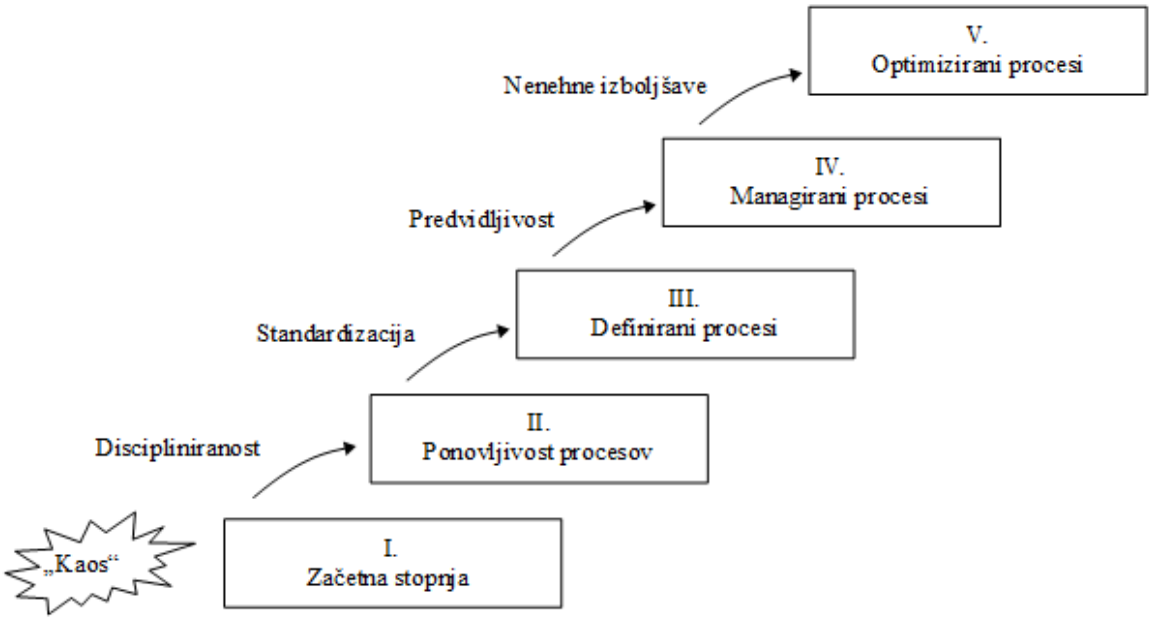
Z uporabo tega modela (Paulk, Curtis, Chrissis, & Weber, 1993) lahko podjetja razvrstimo na pet ravni organizacijske zrelosti:

- **stopnja 1**: začetna faza podjetja (kaotično delovanje, nedefinirani procesi);
- **stopnja 2**: ponovljivi procesi;
- **stopnja 3**: definirani procesi;

- **stopnja 4**: managirani procesi;
- **stopnja 5**: optimizirani procesi.

Ker je bil CMM model v prvi vrsti razvit za podjetja, ki so dobavljala programsko opremo ameriškim vladnim organizacijam, je bil na račun tega deležen precej kritik (npr. Bach, 1999; Wiegers, 1996).

Slika 2: Shema ravni organizacijske zrelosti po CMM modelu



Kritiki modela namreč trdijo, da se razvojne industrije ne da primerjati z manufakturo in, da doseganje višjih stopenj organizacijske zrelosti neke organizacije še ne pomeni, da je programska oprema, ki jo organizacija proizvede, tudi zares dobra. Grady (1992, str. 5) navaja, da je raven organizacijske zrelosti po nadgrajenem CMM modelu (v nadaljevanju CMMI[2]) vseeno zelo dober indikator tega, kakšna je lahko pričakovana verjetnost pojava tveganj pri projektih in prav tako, kakšna je stopnja verjetnosti uspešnega obvladovanja tveganj oz. priložnosti (Tabela 1).

Pri tem izhaja iz dejstva, da je za učinkovito obvladovanje tveganj potrebna metrika, na podlagi katere ugotavljamo in ovrednotimo stanje projekta in se s tem prej in učinkoviteje odzivamo na pojav tveganj (in priložnosti). Ker metrika na nižjih ravneh organizacijske zrelosti še ni razvita, je za pričakovati, da tveganja in njihov vpliv ne bodo ustrezno identificirana in ovrednotena, kar ima za posledico neustrezno ali prepozno odzivanje.

[2] CMMI (*Capability Maturity Model Integration*) model je naslednik teoretičnega CMM modela, prilagojenega za uporabo pri procesih razvoja programske opreme.

Tudi Hopkinson (2011, str. 3–16) gradi svoj RMM model (angl. *Risk maturity model*) na principih CMMI modela, saj trdi, da obstaja pri projektih neposredna povezava med organizacijsko zrelostjo in zrelostjo managementa tveganja in da je od ravni organizacijske zrelosti odvisno, kako učinkovito bomo identificirali in obvladovali tveganja med potekom projekta.

Tabela 1: Ključna področja tveganja po CMMI modelu

Raven org. zrelosti po CMMI	**Procesne značilnosti**	**Ključna področja tveganja**	**Rezultat**
Optimizirani procesi	Nenehno izboljševanje poslovnih procesov	Avtomatizacija procesov	Stopnja produktivnosti in nivo kakovosti / Pričakovana stopnja tveganosti
Managirani procesi	Kvantitativni razvojni proces, institucionalizirana metrika razvojnih procesov	Sledenje spreminjanju tehnologij, analiza problemov, preprečevanje nastajanja problemov	
Definirani procesi	Kvalitativni razvojni proces, institucionalizirani razvojni proces s poudarkom na kakovosti	Merjenje procesov, analiza procesov, kvantitativno planiranje kakovosti	
Ponovljivi procesi	Intuitivni razvojni proces, odvisen od posameznikov	Izobraževanje, tehnične prakse (npr. pregledovanje, sistematsko preizkušanje)	
Začetna stopnja	Ad hoc, kaotični razvojni proces	Management projektov, planiranje projektov, management konfiguracij in verzij, zagotavljanje kakovosti	

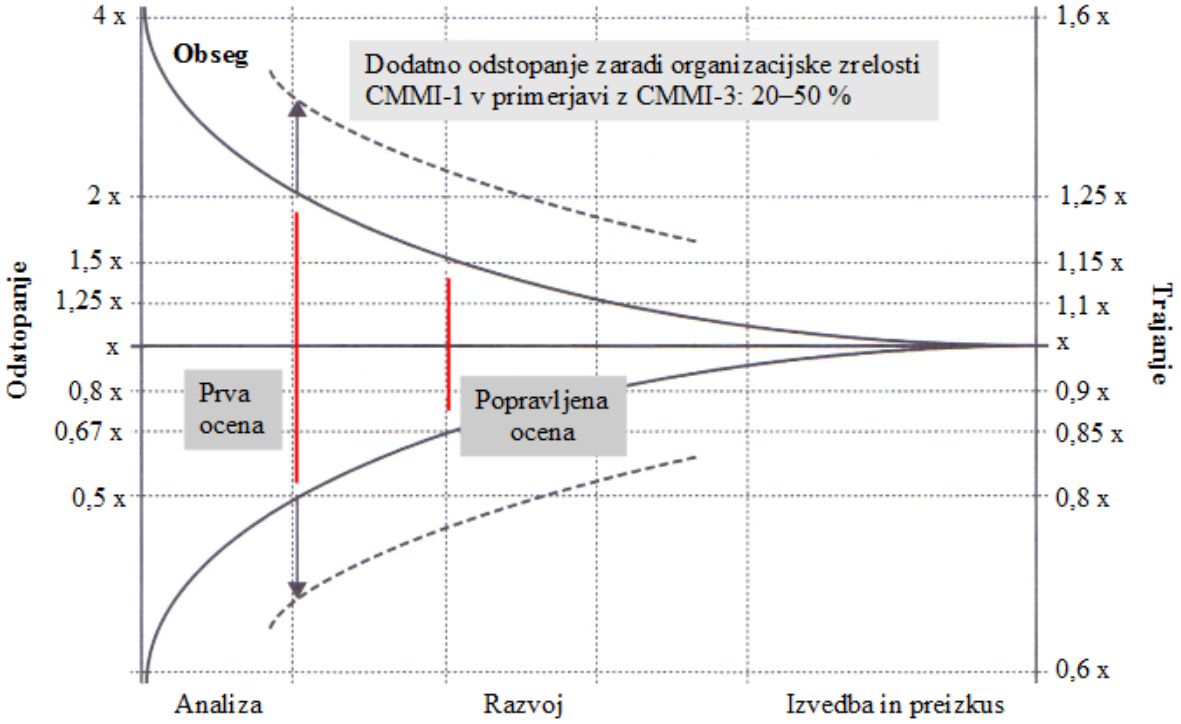
Vir: R. Grady, Practical Software Metrics for Project Managament and Process Improvement, 1992, str. 5.

1.3 Tveganja zaradi neustreznih ocen dela in stroškov

Ebert (2012, str. 152–157 in 294–297) povezuje negotovost pri oceni stroškov in trajanjem projekta v posameznih fazah z organizacijsko zrelostjo projektne organizacije. Pri tem navaja, da negotovost v ocenah in s tem tveganje prekoračitve stroškov in trajanja projekta ni odvisno le od trenutne faze projekta, ampak da dodatno negotovost v ocene vnaša tudi stopnja organizacijske zrelosti. Ebert ocenjuje, da je negotovost v ocenah pri podjetjih na 1. stopnji CMMI modela v primerjavi z organizacijami, ki so na 3. stopnji višja za 20–50 % (Slika 3).

Tudi De Marco in Lister (2003, str. 91–94) ugotavljata na podlagi ugotovitev iz prakse, da je pri organizacijah na 3. stopnji CMMI modela standardno začetno odstopanje ocene stroškov projekta vsaj 30 %. Če strnemo trditve prej omenjenih avtorjev, lahko na 3. stopnji organizacijske zrelosti s 50 % verjetnostjo pričakujemo uresničitev tveganja, da bodo stroški ali časovni rok projekta prekoračeni vsaj za 30 %.

Slika 3: Točnost ocenjevanja v odvisnosti od faze projekta in ravni organizacijske zrelosti



Vir: C. Ebert, Systematisches Requirements Engineering, 2012, str. 153.

Če se vrnemo na uvodoma omenjene ugotovitve McManus-a (2004, str. 2–3) in rezultate raziskav Standish Group-a v letih od 1995 pa vse do danes in upoštevamo, da je 67 % organizacijsko zrelih podjetij, ki razvijajo programsko opremo po raziskavah CMMI Instituta (Slika 4) na 3. stopnji (Moore, 2015, str. 14), lahko sklepamo, da je prekoračitev stroškov razvojnih projektov med 30–200 % pravzaprav pričakovana, standardna vrednost.

Slika 4: Razvrstitev anketiranih razvojnih organizacij po CMMI modelu

Vir: D. Moore, Maturity Profile January 1, 2007 – June 30, 2015, 2015, str. 14.

Iz tega lahko sklepamo, od kje izvira pravilo oz. napotek (angl. *Rule of thumb*), da managerji razvojnih projektov programske opreme, zaradi negotovosti ocen potrebnih stroškov ali obsega dela, v praksi planirajo vsaj 30 % projektne rezerve (angl. *Scheduling reserve*).

Pri tem je potrebno poudariti, da se začetno odstopanje ocene projekta in iz prakse priporočene rezerve nanaša zgolj na stroške in rok izvedbe in ne vključuje drugih tveganj (npr. tehnoloških tveganj). Razlog za to leži, kot smo omenili v uvodu, v tem, da se razvojnih projektov programske opreme ne da primerjati s klasičnimi projekti, saj ni mogoče vnaprej predvideti, koliko kode bo potrebno napisati in koliko preizkušanja bo potrebno za uspešno dokončanje projekta.

1.4 Tveganja povezana z vrstami razvojnih projektov

Po definiciji Project Management Institut-a, v nadaljevanju PMI (2008, str. 5), je projekt začasen podjem, katerega cilj je ustvariti produkt, storitev ali nek rezultat.

Začasna narava projekta pomeni, da ima projekt začetek in konec in da je konec dosežen, ko je realiziran cilj. Stare (2010, str. 17–19) pojem projekta razčlenjuje bolj podrobno in povzema, da je projekt enkraten, časovno in finančno omejen in ciljno usmerjen kompleksen proces logično povezanih aktivnosti.

Za identifikacijo virov tveganj v povezavi z vrsto projekta se lahko poslužimo „matrične“ razvrstitve projektov (Kuster et al., 2011, str. 6–7) in sicer na:

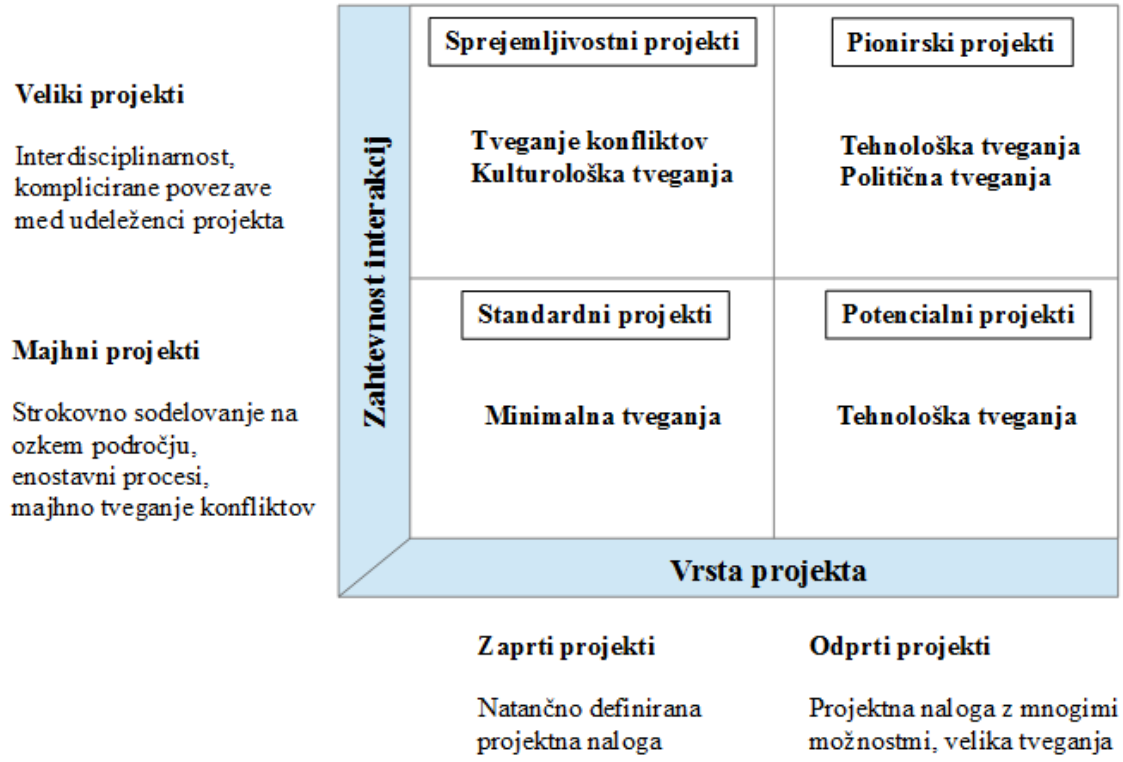
1. **Standardne projekte**, ki se lahko odvijajo na enostaven in standardiziran način. Projektna naloga (npr. razvoj enostavne aplikacije) je enostavna in ni veliko možnosti za nastopanje konfliktov med udeleženci. Ker gre za ustaljene postopke in znana področja, tveganj praktično ni ali pa so minimalna.
2. **Sprejemljivostne projekte**, za katere je značilno, da je kljub enostavni projektni nalogi potrebna intenzivna interakcija med naročniki projekta, internimi udeleženci (izvajalci) in zunanjim okoljem projekta. Pri teh vrstah projektov nastopajo tveganja predvsem pri komunikaciji in razumevanju naloge in vplivov projekta na zunanje okolje. Potek informacij mora biti v vseh smereh nemoten in jasen, da bi s tem preprečili tveganje konfliktov ali polariziranje udeležencev tekom projekta. Še posebej so pri tovrstnih projektih izražena kulturološka tveganja v primeru, da udeleženci projekta prihajajo iz različnih kulturnih okolij.

3. **Potencialne projekte**, to so projekti z velikim potencialom in pri katerih nastopa ogromno neznank, ampak so po svoji naravi vseeno zaključeni in pri njih ni mnogo interakcij niti med notranjimi niti med zunanjimi udeleženci. Tveganja, ki nastopajo pri teh projektih, so omejena predvsem na ustreznost rešitev oz. tehnološka tveganja. V to kategorijo spadajo predvsem pred-projekti, študije izvedljivosti in raziskovalni projekti.
4. **Pionirske projekte**, to so projekti z velikim potencialom in hkrati z izrazito interakcijo med notranjim in zunanjim okoljem. Pri teh projektih pogosto premikamo meje mogočega. V ustaljene strukture vnašamo nove postopke ali procese in spreminjamo obstoječi način dela. Tveganja, ki nastopajo pri pionirskih projektih, so predvsem tehnološka in politična.
   Pri tehnoloških gre za tveganja, povezana predvsem z še neznanimi tehnologijami, za katere ne moremo oceniti, kakšne težave nam bodo povzročale pri izvedbi projekta. Pri političnih tveganjih gre predvsem za tveganje notranjih konfliktov med udeleženci projekta. Zaradi uvajanja novih tehnologij in procesov se lahko pojavi polarizacija pri izbiri tehnologije, saj udeleženci projekta težijo k uporabi preizkušenih rešitev in se upirajo vpeljavi novih in neznanih, razen, če to ni nujno potrebno.

Po »matričnem modelu« (Slika 5) bi projekte razvoja vgrajene programske opreme najlažje umestili med projekte z visoko medsebojno interakcijo med udeleženci projekta in zunanjim okoljem projekta, torej med sprejemljivostne in pionirske projekte.

Pri tveganjih bodo tako prevladovala tehnološka tveganja, kar je tudi pričakovano, kulturološka (npr. v primeru virtualnega tima programerjev z različnih celin) in politična, ki jih managerji projektov razvoja programskih rešitev v praksi največkrat zanemarijo.

Slika 5: Vrste projektov glede na zahtevnost interakcij med udeleženci projekta



Vir: J. Kuster et. al., Handbuch Projektmanagament, 2011, str. 6.

1.5 Tveganja povezana z modeli razvoja programske opreme

V strokovni literaturi in spletnih virih najdemo veliko število modelov razvoja programske opreme, ki jih je razvila stroka v zadnjih 20-tih letih, skladno s potrebami, spoznanji in spreminjanjem tehnologije.

Modele razvoja lahko razdelimo po njihovih lastnostih na tri glavne skupine:

- **linijske** razvojne modele;
- **ciklične** razvojne modele;
- **inkrementalne/evolucijske** modele.

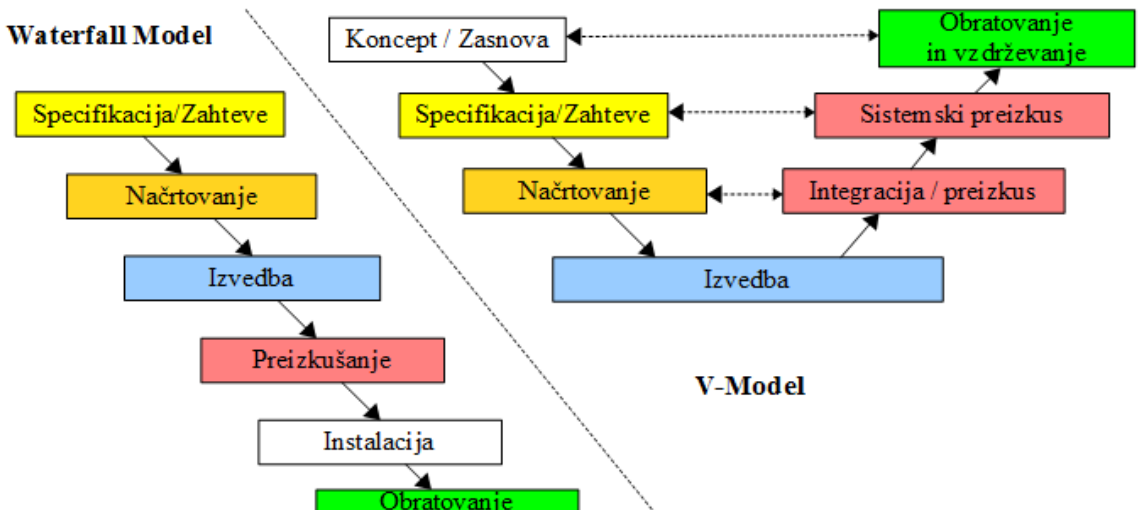
Vseh razvojnih modelov ne bomo podrobno opisovali, ampak bomo v nadaljevanju opisali le najznačilnejše predstavnike, njihove značilnosti in kakšna tveganja se pojavljajo pri posameznih vrstah razvojnih modelov.

1.5.1 Linijski razvojni modeli

Za linijske razvojne modele, ki so najstarejši razvojni modeli programske opreme in izhajajo iz strojništva, je značilno, da faze projekta potekajo zaporedno, druga za drugo.

Pri Waterfall modelu (Royce, 1970) si posamezne faze razvoja sledijo druga za drugo (Slika 6), medtem ko V-model (IABG, 2006) logiko linijskega (Waterfall) modela dopolnjuje tako, da zrcali posamezne faze razvoja s preizkušanjem oz. validacijo. Pri V-modelu načrtujemo preizkus vseh izvedbenih elementov že v fazi načrtovanja (Slika 6).

Slika 6: Shematski prikaz Waterfall in V-modela



Pri linijskih razvojnih modelih je poudarek predvsem na začetni fazi, pri postavitvi projektnih zahtev in natančnem planiranju, ki naj bi preprečevalo pojav tveganj v poznejših fazah projekta. Značilnost teh modelov je, da prehajamo med posameznimi fazami šele, ko je predhodna faza v celoti zaključena. Ker je pri linijskih razvojnih modelih velik poudarek na podrobni projektni dokumentaciji, še posebej v začetni fazi oblikovanja projektnih zahtev in skozi celoten razvojni proces, so modeli primerni šele takrat, ko so v podjetju procesi formalizirani. To pomeni, da mora podjetje po CMMI modelu biti vsaj na 2.–3. ravni poslovne zrelosti.

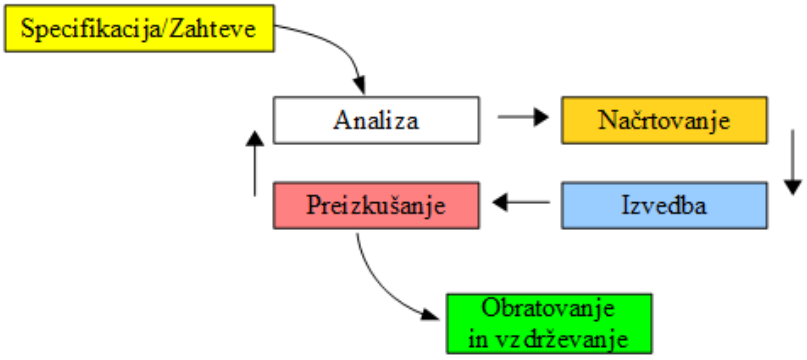
Največji problem, ki se pojavlja pri linijskih modelih razvoja programske opreme, je **tveganje sprememb** v zahtevah naročnikov projekta. Vsaka sprememba namreč povzroča odstopanje od projektnega plana in s tem zaostanke, v najslabšem primeru pa vračanje na začetek projekta, k ponovnemu definiranju in prilagoditvi zahtev.

Čeprav je osnova linijskih modelov dobra projektna dokumentacija, ki zagotavlja transparentnost in s tem ugodno vpliva na odpravo tveganj, je težava v tem, da v razvojnem timu nujno potrebujemo ločene skupine ljudi, ki se ukvarjajo s specifikacijami, izvedbo in preizkušanjem, kar pa ni vedno mogoče. V nasprotnem primeru linijskega modela ne moremo učinkovito aplicirati in s tem izničimo njegove prednosti (**tveganje pristranskosti in kakovosti**). Pri tem moramo upoštevati tudi **tveganje časovnega zaostanka** projekta, saj se nam lahko zgodi, da zaradi predolge faze natančnega specificiranja in izvedbene faze, preizkusno fazo stisnemo na minimum in s tem sprožimo pojav sekundarnih tveganj (**tveganje kakovosti**).

1.5.2 Ciklični razvojni modeli

Ciklični razvojni modeli temeljijo na ideji Deming-ovega modela nenehnih izboljšav (v nadaljevanju PDCA), saj neko tehnično rešitev najprej planiramo (angl. *Plan*), izvedemo (angl. *Do*), preverimo, ali ustreza tehničnim in naročnikovim zahtevam (angl. *Check*) in ugotovimo, kaj je potrebno spremeniti (angl. *Act/adjust*). Nato začnemo nov ciklus s planiranjem izvedbe popravkov in cikel ponavljamo, dokler projekt ni zaključen (Moen, 2010). Tipičen predstavnik cikličnih modelov je Rapid Aplication Development (v nadaljevanju RAD) model (Slika 7).

Slika 7: Shematski prikaz RAD razvojnega modela



Pri RAD modelu skozi celoten potek razvoja programske opreme ponavljamo PDCA cikel in sicer tako dolgo, dokler v fazi preizkušanja ne dobimo zadovoljivega rezultata (kakovosti).

Ciklični razvojni modeli so primerni predvsem za podjetja na nižjih stopnjah zrelosti po CMMI modelu, ne glede na to, ali imajo formaliziran projektni management ali so šele na čistem začetku (npr. v primeru razvoja novih produktov), saj ugodno vplivajo na obvladovanje **tveganja sprememb**. Tveganja, ki se pojavljajo pri teh projektih, so predvsem tveganja **uspešnega zaključka projekta**, saj lahko zaradi nenehnega spreminjanja prvoten cilj popolnoma zgrešimo in **tveganja prekoračitve stroškov** (v primeru, da potrebujemo za dokončanje projekta preveč iteracij).

1.5.3 Inkrementalni razvojni modeli

Inkrementalni razvojni modeli programske opreme so kombinacija obeh prej opisanih modelov. Pri teh modelih gre za postopni razvoj programske opreme v definiranih korakih (inkrementih).

Osnovna ideja je, da se znotraj inkrementa projektne zahteve ne spreminjajo. Če se projektne zahteve vmes vseeno spremenijo, začnemo nov (zaključen) razvojni proces, ki ga z omejenim dodajanjem novih funkcijskih sklopov ali sprememb inkrementalno ponavljamo, dokler niso izpolnjene vse projektne zahteve (npr. Graham, 1990). Inkrementalnega modela se po večini poslužujemo, kadar:

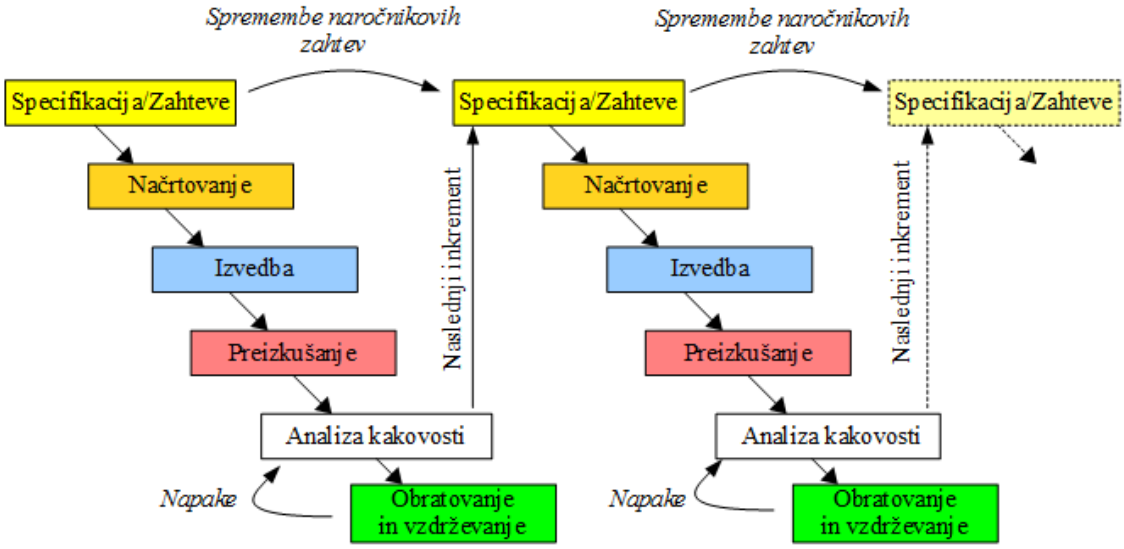
1. ocenimo, da ima projekt visoko **tveganje spremembe naročniških zahtev**;
2. ocenimo, da je velika verjetnost napak ali funkcijskih pomanjkljivosti, ki bi jih ugotovili šele med obratovanjem ali po zaključku (**tveganje zaključka projekta ali kakovosti**).

Tipični predstavniki inkrementalnih razvojnih modelov so: linijski inkrementalni model (Slika 8) in razvojni procesi, ki temeljijo na agilnem managementu (Slika 9). Prednost inkrementalnih razvojnih modelov je v lažjem predvidevanju in obvladovanju tveganj, saj se v posameznih inkrementih projektne zahteve ne spreminjajo (so „zamrznjene“). Na ta način

dosežemo, da so spremembe med posameznimi inkrementi znane in morebitna tveganja lažje predvidimo in obvladujemo. Slabosti linijskega inkrementalnega razvojnega modela se kažejo predvsem v tem, da se zaradi majhnih in omejenih korakov projekti odvijajo zelo počasi in tvegamo, da ob zaključku rezultat projekta ne bo več aktualen (angl. *Time to market*) oz. bo zastarel (**tveganje zastarelosti rezultata projekta**).

Dodatno tveganje pri tovrstnih projektih je, da z napredovanjem z majhnimi in omejenimi koraki naročnikom nehote sugeriramo delovanje po načelu najmanjšega odpora (angl. *Minimum effort*), kar ima za posledico zelo visoko **verjetnost tveganja predčasne prekinitve projekta** ali **nezadovoljstva**, saj se zaradi **tveganja slabega zaključka projekta** ali **tveganja prekoračitve stroškov** naročniki zadovoljijo z manjšim funkcijskim obsegom od prvotno načrtovanega (Blank, 2014).

Slika 8: Shematski prikaz linijskega inkrementalnega razvojnega modela

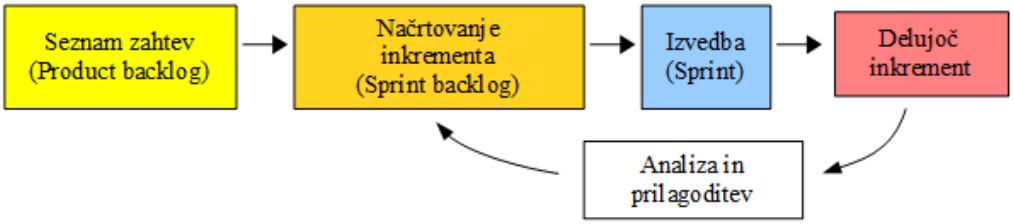


Pri razvojnih projektih vgrajene programske opreme je še posebej visoka verjetnost tehnoloških tveganj. Pri poznejših inkrementih se namreč pogosto dogaja, da prvotna rešitev in nadgraditve niso več ustrezni za implementacijo prvotno zahtevanih funkcij (**tveganje neustreznosti**). Razvojni model z dolgimi faznimi poteki in majhnimi inkrementi je zato primeren le za organizacije na višjih stopnjah zrelosti po CMMI modelu, ki razvijajo varnostno kritične vgrajene sisteme (angl. *Safety critical embedded systems*), npr. za letalsko, medicinsko ali vesoljsko opremo.

Obratno smo pri cikličnih inkrementalnih razvojnih procesih, ki izhajajo iz agilnega managementa (Slika 9), namenoma usmerjeni v učinkovito delovanje razvojnega tima in prilagoditve izdelka, s katerim si želimo čim hitreje zagotoviti tržni delež[3] in konkurenčno prednost. Zato namenoma zanemarjamo proces dokumentiranja in formaliziranja postopkov (npr. Fowler, 2001).

[3] Pri tem mislimo predvsem lansiranje izdelkov, ki izpolnjujejo minimalne zahteve uporabnikov (angl. *Minimum viable product*).

Slika 9: Shematski potek cikličnega inkrementalnega modela (Agile)



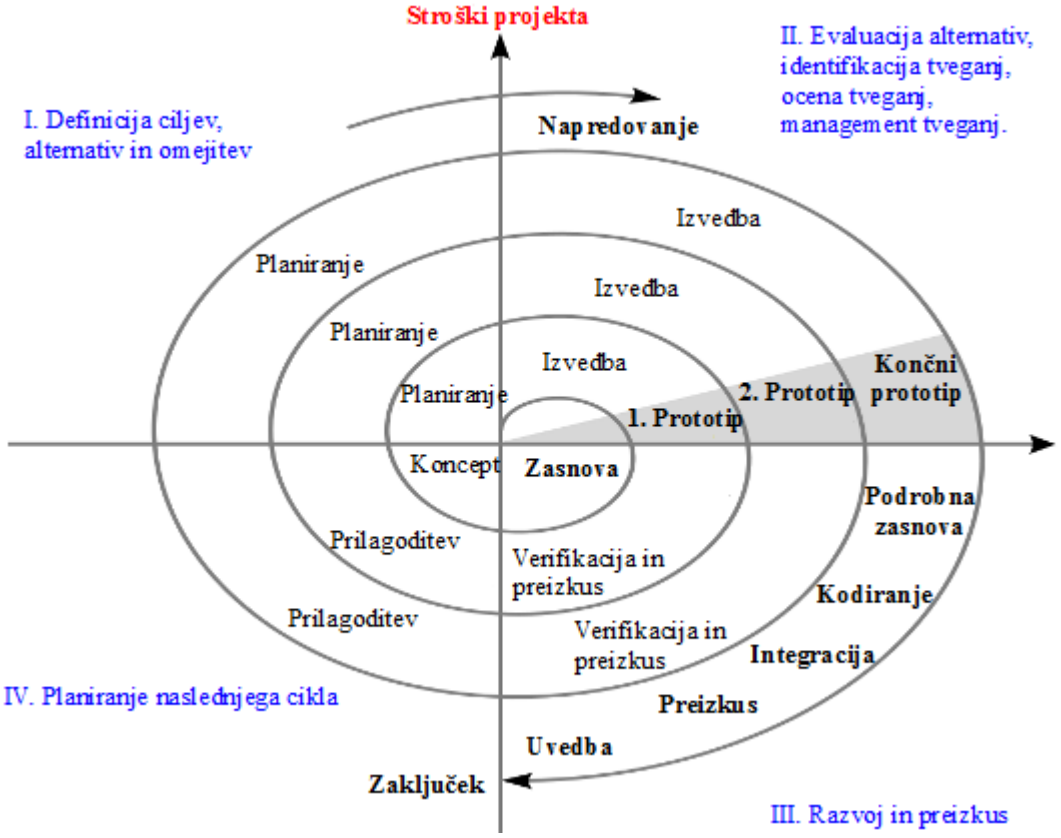
S tem tvegamo, da bo kakovost končnega izdelka (**tveganje kakovosti**) zadovoljujoča šele po mnogih iteracijah, kar lahko izniči prednosti hitrega razvoja (**tveganje nezadovoljstva kupcev**).

1.5.4 Spiralni razvojni model

S stališča obvladovanja tveganj je najprimernejši Boehm-ov (1989, str. 39) razvojni model. Model združuje vse prej opisane razvojne modele, ki jih med potekom projekta smiselno uporabljamo in s tem minimiziramo tveganja, ki izhajajo iz posameznih razvojnih modelov (Slika 10). Boehm-ov razvojni model temelji na štirih stopnjah posameznega projektnega cikla:

1. **definicija** projektnih ciljev, alternativ in omejitev;
2. **evaluacija** alternativ, identifikacija, ocena in aplikacija metod za zmanjševanje tveganj;
3. **izvedba** in preizkus vmesnega izdelka;
4. **planiranje** naslednjega cikla in nadaljevanje projekta.

Slika 10: Boehm-ov spiralni razvojni model



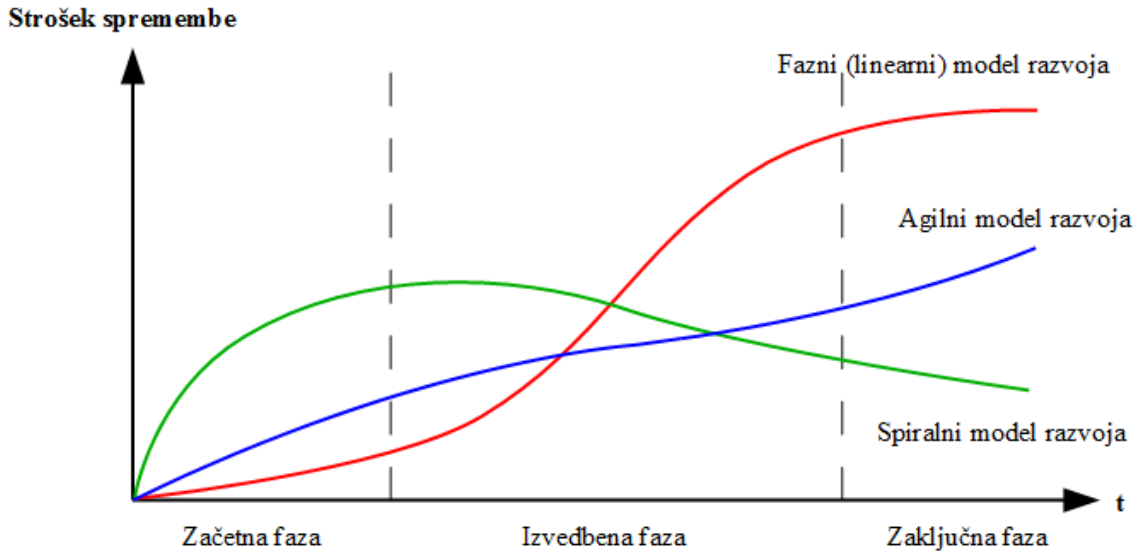
Vir: B. Boehm, Software Risk Management, 1989, str. 43.

Bistvo Boehm-ovega modela je, da temelji na obvladovanju tveganj (angl. *Risk driven model*). Kot smo omenili v prejšnjih poglavjih, so v večini primerov največja tveganja povezana s tveganjem sprememb (Stare, 2010, str. 4–5), še posebej v poznejših fazah projekta (Slika 11). Boehm-ov spiralni model omogoča, da v začetnih fazah prehajamo spiralno skozi štiri stopnje modela. V začetnih krogih se poslužujemo cikličnih razvojnih modelov, s katerimi lažje obvladujemo **tveganja sprememb** ali **tehnološka tveganja** kot s faznimi razvojnimi modeli. Ko začetni razvoj stabiliziramo do te stopnje, da dosežemo funkcijski, tehnološki, finančni in časovni konsenz, preidemo na fazne razvojne modele in projekt hitro in kakovostno zaključimo. Če to ni dovolj, v poznejših krogih lažje inkrementalno nadaljujemo, vse dokler cilji projekta niso dokončno izpolnjeni.

S takšnim pristopom se stroškovno učinkovito izognemo poznejšim spremembam, saj stabiliziramo potek projekta že v začetni fazi, ko skupni stroški projekta še niso tako visoki in si glede na obseg izvedenih del še lahko privoščimo spremembe (Slika 11). Boehm-ov spiralni model vključuje tudi zelo pomembno komponento, predčasno prekinitev projekta na prehodu med 3. in 4. kvadrantom. Tako projekt prekinemo, preden bi ga prekinili zaradi nastopa **tveganja znatne prekoračitve stroškov** ali **prevelikih časovnih zaostankov**.

Z modelom zelo dobro obvladujemo tudi **tehnološka tveganja**, saj se z novimi tehnologijami spoznavamo v začetnih (prototipnih) ciklih. Tako odpori do novih tehnologij in politična tveganja ne pridejo do izraza.

Slika 11: Strošek sprememb v odvisnosti od faze projekta in razvojnega modela



Vir: A. Stare, Obvladovanje sprememb v izvedbi projekta, 2010, str. 5; S. W. Ambler, Examining the Agile Cost of Change Curve.

Kot navaja Boehm (1989, str. 34–35), je težava modela s stališča aplikativnega razvoja programske opreme med drugim, da ga je težko aplicirati na naročniške projekte, zato je model primernejši za produktni razvoj aplikacij, torej za trg. Nadalje Boehm (1989, str. 35) navaja, da je problem tudi v tem, da se v praksi pri oceni tveganj pogosto zanašamo na ekspertizo posameznikov, saj v proces analize tveganj praviloma vključujemo specialiste z bogatimi izkušnjami. S tem tvegamo, da bodo izvajalci, zaradi pomanjkljivih izkušenj, s strani ekspertov identificirana tveganja slabo razumeli. Nasprotno eksperti ne potrebujejo tako podrobnih specifikacij kot manj izkušeni razvijalci, kar pomeni, da morebitna tveganja ne bodo upoštevana. Na tem mestu se pojavi problem **tveganja napačnih odločitev** (angl. *Design/implementation decision driven risk*), ki nas lahko mimogrede zapelje v novo korekturno spiralo, s tem pa poveča stroške in izvedbeni čas projekta.

Tveganje napačnih odločitev, v povezavi s podrobnostjo specifikacij, se po izkušnjah vedno pojavlja pri projektih, pri katerih v projekt vključujemo zunanje izvajalce. V primeru težav zaradi neustrezne izvedbe oz. odločitve se zunanji izvajalci zelo radi sklicujejo na pomanjkljivo specifikacijo. Ker se zunanjih izvajalcev poslužujemo le v primerih, ko imamo pomanjkanje virov, oz. bi bila lastna izdelava predraga, tudi ne moremo zagotoviti „super podrobne“ specifikacije. Slednje bi zahtevalo toliko časa in lastnih virov, da se zunanjih virov sploh ne bi več splačalo vključevati (**tveganje zunanjih virov**).

Na problem planiranja zunanjih virov in pomembnost ocene stroškov izvedbe, v povezavi z uporabo lastnih ali zunanjih virov, opozarja tudi Stare (2011a). Nižja cena zunanjih virov namreč prav zaradi povečanega tveganja ne vpliva nujno ugodno na finančni izid projekta, saj

moramo upoštevati tudi oportunitetne stroške, kar pomeni, da je lahko uporaba zunanjih virov dražja od uporabe lastnih. Pri tem je nujno potrebno upoštevati še **produktivnost zunanjih** izvajalcev, saj ta neposredno vpliva na tveganje **časovnih zaostankov**, predvsem zato, ker v začetnih fazah sodelovanja po navadi nimamo podatkov o produktivnosti.

1.5.5 Povzetek razvojnih modelov in z njimi povezanih tveganj

Tabela 2: Povzetek razvojnih modelov in z njimi povezanih tveganj

Vrsta modela	Prednosti	Slabosti	Specifična tveganja
Linijski modeli	- Sosledje faz; - definiran prehod med fazami; - načrtovanje in dokumentacija; - formaliziran proces.	- Togost; - pojav sprememb v poznejših fazah razvoja; - potrebnih več virov.	- Tveganje sprememb; - tveganje napačnih odločitev v začetku; - tehnološka tveganja.
Ciklični modeli	- Hitri cikli; - prilagoditve spremembam; - preizkušanje ustreznosti rešitev.	- Pomanjkljiva dokumentacija.	- Tveganje zgrešitve prvotnega cilja.

se nadaljuje

nadaljevanje

Inkrementalni modeli	- Sosledje faz; - ni sprememb zahtev med posameznimi etapami; - analiza in načrtovanje pred naslednjo etapo; - obvladovanje tveganj z majhnimi spremembami (inkrementalni model); - obvladovanje uporabniških in tržnih zahtev s hitrimi prilagoditvami (agile).	- Počasnost (razen pri agile); - potrebno natančno načrtovanje in dokumentiranje med etapami; - potreben management sprememb.	- Tveganje zastarelosti izdelka ob zaključku projekta; - tveganje tehnološke neustreznosti v poznejših etapah.
Spiralni modeli	- Uporaba cikličnih metod v začetku; - stabiliziranje zahtev v začetnih fazah; - analiza in načrtovanje naslednje faze; - predčasna prekinitev projekta;	- Napačne (ekspertne) analize nas lahko zapeljejo v nepotrebne cikle in s tem povečajo strošek projekta.	- Tveganje napačnih odločitev na podlagi ekspertnih ocen.

	- obvladovanje teh. tveganj; - obvladovanje političnih tveganj.		

2 PROCES MANAGEMENTA TVEGANJ

PMI (2008, str. 273) definira management projektnih tveganj kot proces, ki poteka v naslednjih fazah:

1. **planiranje managementa tveganj** – proces, s katerim definiramo aktivnosti v zvezi z managementom tveganj pred začetkom projekta, torej ali bomo management tveganj izvajali ali ne, v kakšni obliki in v kakšnem obsegu;
2. **identifikacija tveganj** – proces ugotavljanja možnih tveganj in dokumentiranje vrst tveganj;
3. **kvalitativna analiza tveganj** – proces, s katerim identificirana tveganja razvrstimo po verjetnosti dogodka in velikosti vpliva na projekt;
4. **kvantitativna analiza tveganj** – proces numerične ocene vpliva posameznih tveganj na finančni rezultat projekta;
5. **planiranje odzivov** – proces razvoja opcij in aktivnosti, ki jih bomo izvajali v primeru, ko bo prišlo do tveganega dogodka, ki smo ga identificirali in ovrednotili v prejšnjih fazah in s tem minimizirali negativen vpliv na rezultat projekta;
6. **kontroliranje tveganj** – proces implementacije managementa tveganj v skladu s planom. V tem delu sledimo pojavu in odpravljamo tveganja, identificiramo nova tveganja in spremljamo učinkovitost metod za obvladovanje tveganj.

Pri tem je potrebno poudariti, da čeprav ves čas govorimo o **negativnih tveganjih**, torej o tveganjih, ki slabšalno vplivajo na rezultat projekta, velja enako tudi za pozitivna tveganja, tj. **priložnosti**, ki lahko pozitivno vplivajo na izid projekta in jih pri managementu tveganj ne smemo spregledati. Prav tako je pomembno, da proces managementa tveganj razumemo in sprejmemo kot **iterativni proces**, ki se odvija in spreminja ves čas tekom projekta (PMI, 2008, str. 276).

2.1 Vrste in viri tveganj

Mulcahy-jeva (2010, str. 13) navaja naslednje osnovne vrste tveganj:

1. **poslovno tveganje** – tveganje negativnega (ali pozitivnega) poslovnega izida**;**

2. **čisto tveganje** – tveganje čiste izgube, npr. zaradi požara, zloma ali nesreče. To vrsto tveganj lahko tudi zavarujemo in s tem prenesemo na drug subjekt.

Po Mulcahy-jevi management tveganj primarno obsega ugotavljanje naslednjih faktorjev tveganja:

1. **verjetnost** – da se bo nek tvegan dogodek (ali priložnost) zgodil;
2. **vpliv oz. posledica** – za projekt v primeru, da se tvegan dogodek zgodi;
3. **čas pojava** – kdaj med potekom projekta se tvegan dogodek lahko zgodi;
4. **pogostost pojavljanja** – kolikokrat se lahko tvegan dogodek med potekom projekta ponovi.

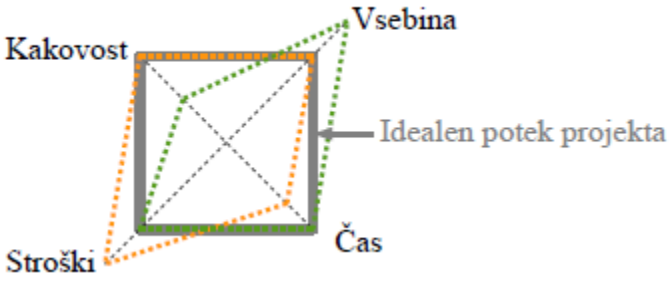
Pri času pojava tveganja gre za oceno vpliva tveganja v točno določenem trenutku (npr. na začetku ali koncu projekta), medtem ko gre pri pogostosti pojavljanja za oceno vpliva, če se bo neko tveganje ponovilo večkrat ali manjkrat kot bi sicer pričakovali (npr. število deževnih dni).

Tvegan dogodek oz. tveganja se vedno pojavljajo v prihodnosti. Če se pojavijo, vplivajo vsaj na eno dimenzijo projekta (PMI, 2008, str. 275):

1. na **vsebino** projekta – ali bodo izpolnjene vse projektne zahteve;
2. na **časovni potek** projekta – ali bo projekt zaključen v predvidenem roku;
3. na **stroške** projekta – ali bo projekt zaključen v okviru predvidenih stroškov;
4. na **kakovost** rezultata – ali bo rezultat projekta izpolnil pričakovani nivo kakovosti.

Sprememba katerekoli od štirih dimenzij projekta, zaradi pojava tveganja, vpliva tudi na ostale dimenzije projekta. Razmerje med dimenzijami projekta prikazuje t. i. Sneedov »hudičev kvadrat« (Slika 12), katerega ideja je, da je ploščina kvadrata vedno enaka, ne glede na to, kako premikamo vogale.

Slika 12: Sneedov hudičev kvadrat

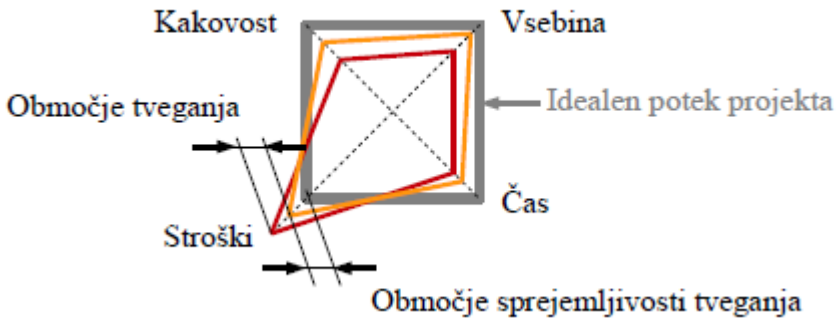


Vir: A. Berg, The Magic Triangle and Devil's Quadrangle – Understanding Project Management Models, 2015.

Tveganja vplivajo na posamezne dimenzije projekta s pozitivnimi in negativnimi silami in jih pri tem premikajo znotraj ali izven prvotno načrtovanih mej sprejemljivosti (Slika 13).

S stališča managementa tveganj je že na samem začetku projekta potrebno določiti okvire spremenljivosti (angl. *Risk tolerance area*) za posamezna tveganja in njihov vpliv. Gre za to, da lahko določeno stopnjo tveganja sprejmemo kot običajno projektno tveganje. Dokler se tveganja pojavljajo znotraj določenih mej, se z njimi posebej ne ukvarjamo.

Slika 13: Območje tveganja in sprejemljivosti tveganja



Pri razvojnih projektih se tveganjem ne moremo popolnoma izogniti in jih tudi ne moremo v celoti vnaprej predvideti, saj bi to zahtevalo preveč časa in energije. Zato je potreben kompromis med energijo, ki jo vložimo v proces in učinkom, ki ga pri tem dosežemo. Tako poskušamo zagotoviti pravo razmerje (angl. *Trade-off*) med posameznimi cilji projekta (PRINCE2, str. 240).

V praksi to pomeni, da se zavestno odločimo, da bomo imeli pri projektu opravka s preostalim tveganjem (angl. *Residual risk*), ki ga, ali nismo mogli v celoti vnaprej predvideti ali pa v celoti odpraviti (Project Management Knowledge, 2010).

DeMarco in Lister (2003, str. 102) navajata naslednja ključna tveganja, ki se pojavljajo pri razvojnih projektih programske opreme[4]:

1. **prekoračitev časa trajanja projekta** (napačna ocena obsega potrebnih del);
2. **inflacija zahtev** (spreminjanje in dodajanje novih zahtev tekom projekta);
3. **fluktuacija človeških virov** (odpoved in zamenjava ključnih izvajalcev);

[4] Pri tem imata avtorja v mislih predvsem IT projekte oz. razvoj aplikativne programske opreme, vendar velja enako tudi za razvojne projekte vgrajene programske opreme.

4. **zlom specifikacij** (odpoved pogajalskega procesa in konsenza o načinu izvedbe);
5. **neučinkovitost** (manjša produktivnost od predvidene).

Tudi različni avtorji, npr. Boehm (2000, str. 94–96), McManus (2004, str. 41), McConnel (1996), Koopman (2011, str. 1–43), Arnuphaptrairong (2011, str. 732–737), Addison (2002, str. 128–140) in Ropponen (2000, str. 98–112) opisujejo v svojih študijah najpogostejša tveganja pri razvojnih projektih programske opreme.

Omenjeni avtorji opisujejo predvsem tveganja, povezana z razvojem aplikativne programske opreme, ki so jih pridobili na osnovi lastnih izkušenj, anket in preučevanj literature. Izmed vseh edino Koopman (2010, str. 1–5) podrobno navaja tveganja, povezana neposredno z razvojem vgrajene programske opreme.

Ker je težko potegniti ločnico med tveganji, povezanimi z razvojem aplikativne in vgrajene programske opreme, smo pri preučevanju opisanih najpogostejših tveganj, ki jih prej omenjeni avtorji opisujejo, razvrstili v tabelo najpogostejša tveganja (Tabela 3), ki jih lahko uporabimo kot izhodišče za identifikacijo tveganj pri razvojnih projektih vgrajene programske opreme.

Tabela ni popolna, vendar smo v njej poskušali opisati najbolj pomembna tveganja in jih smiselno razdeliti na sedem glavnih področij.

Tabela 3: Najpogostejši viri tveganj pri razvoju vgrajene programske opreme

Področje	**Najpogostejša tveganja**
Funkcijske oz. projektne zahteve	- Pogosto spreminjanje funkcijskih zahtev tekom projekta; - pomanjkljiva analiza funkcijskih zahtev; - pomanjkljiv ali nerazumljiv opis zahtev; - strokovni nivo opisa zahtev je višji od strokovnega nivoja izvajalcev; - predvidevanje, da bodo razvijalci razumeli jezik eksperta; - pomanjkljiva omejitev funkcijskih zahtev (mejnih primerov); - pomanjkljiv opis prioritet in dovoljenih odstopanj (območje sprejemljivega tveganja).
Zasnova programske opreme	- Pomanjkljiva specifikacija sistema; - napačna zasnova sistemskih zahtev; - nekonsistentna zasnova vmesnikov med posameznimi komponentami; - slaba zasnova poteka algoritmov; - uporaba sosledja faz namesto (stanj) končnih avtomatov; - slaba analiza časovnih potekov v realnem času (časovne razvrstitve opravil);

	- preobremenjenost spomina ali procesne enote; - preveč in slabo komentirane globalne spremenljivke; - slab management konkurenčnih procesov; - uporaba rešitev „iz domače garaže“, namesto preizkušenih industrijskih rešitev.
Razvojni proces	- Neformalen proces; - neupoštevanje preteklih izkušenj; - slab ali neobstoječ management verzij; - razvoj ne poteka tako, da bi bila možna ponovna uporaba komponent na drugih projektih; - slaba metrika za „kdaj je dobro dovolj dobro“, še posebej pri uporabniških vmesnikih; - razvojni proces ni prilagojen poteku in rezultatu projekta; - razvijanje funkcij, ki niso nujno potrebne (angl. *Gold plating*).
Organizacija in okolje	- Spremembe v managementu organizacije; - poslovna politika v nasprotju s cilji projekta; - nestabilnost organizacije; - nestabilnost zunanjega poslovnega okolja; - prestrukturiranje organizacije tekom projekta.

se nadaljuje

nadaljevanje

Zahtevnost projekta	- Visoka kompleksnost medsebojnih povezav med posameznimi komponentami sistema; - časovna sinhronizacija med komponentami strojne in programske opreme.
Tehnologija	- Uporaba nove, razvijalcem še neznane tehnologije; - uporaba nezadostno zrele tehnologije (tehnologija se še razvija); - preskok vmesnih prototipnih faz pri vpeljavi nove tehnologije; - predvidevanje, da bo določena tehnologija rešila določen problem brez predhodnega preizkusa (npr. nakup orodja, komponent, prog. knjižnic, intelektualne lastnine).
Planiranje in kontroliranje	- Pomanjkljive izkušnje pri managementu projektov; - nezadovoljiv nadzor projekta; - slabo planiranje potrebnih virov (človeških, finančnih); - slabo planiranje projekta; - pomanjkljiva in slabo planirana komunikacija; - pomanjkljiv management.

Vir: B. Boehm, Project Termination Doesn't Equal Project Failure, 2000, str. 94–96; J. McManus, Risk Management in Software Development Projects, 2004, str. 41; P. Koopman, Risk Areas in Embedded Software Industry Projects, 2010, str. 1–5; P. Koopman, Avoiding the Top 43 Embedded Software Risks, 2011, str. 1–43; T. Arnuphaptrairong, Top Ten Lists of Software Project Risks: Evidence from the Literature Survey, 2011, str. 732–737; T. Addison, Controlling Software Project Risks – an Empirical Study of Methods just by Experienced Project Managers, 2002, str. 128–140; J. Ropponen, Components of Software Development Risk: How to Address Them? A Project Management Survey, 2000, str. 98–112; S. McConnel, Classic Mistakes Enumerated, 1996.

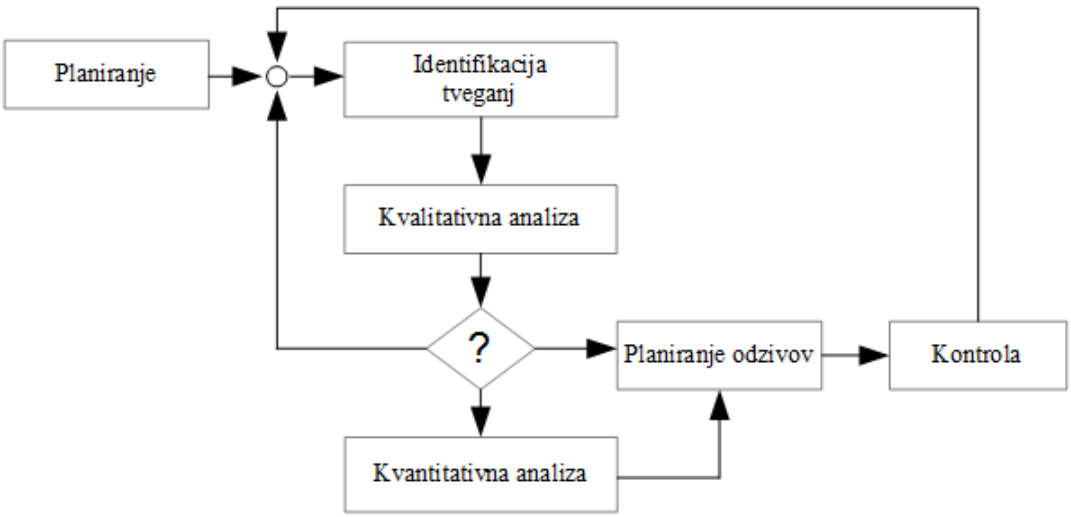
2.2 Izbira procesa managementa tveganj

Različni avtorji opisujejo različne procese managementa tveganj. Menimo, da je proces kot ga opisuje PMI (Slika 14), najbolj ustrezen za aplikacijo pri razvojnih projektih vgrajene programske opreme. Proces se začne s planiranjem managementa tveganj in nadaljuje s postopkom identifikacije tveganj. Po identifikaciji tveganj tveganja kvalitativno ocenimo po verjetnosti nastanka in vplivu na potek projekta.

Če je potrebno, opravimo tudi kvantitativno analizo in numerično ovrednotimo identificirana tveganja. Na manjših projektih se kvantitativne analize ne splača delati, saj je relativno zahtevna.

V tej točki se lahko vrnemo na začetek, na identifikacijo tveganj ali pa nadaljujemo s planiranjem odzivov na identificirana tveganja. Ko imamo izdelan plan odzivov na tveganja, določimo pristojne osebe (angl. *Risk owner*) in začnemo s kontrolo in spremljanjem tveganj med potekom projekta.

Slika 14: Shematski prikaz procesa managementa tveganj



Ker proces ni enkraten, se med potekom projekta po potrebi vedno znova vračamo na začetek. Pri tem kontroliramo:

- ali so tveganja, ki smo jih prvotno identificirali, še vedno aktualna in so pravilno razvrščena po prioriteti;
- ali smo vmes identificirali nova tveganja in kakšen je njihov vpliv;
- ali je prišlo do tveganega dogodka in kakšen je status odprave posledic.

Proces ponavljamo do zaključka projekta. S tem postopkom tudi minimiziramo preostala tveganja.

2.2.1 Planiranje managementa tveganj

Strateški pomen planiranja managementa tveganj na končen rezultat projekta, kot smo povedali uvodoma, opisuje Kerzner (2001, str. 156). Kot prikazuje Slika 15, na končen rezultat projekta vplivata predvsem dva dejavnika, in sicer:

- **tehnična nezmožnost izvedbe in/ali;**
- **slab management tveganj**.

Kerzner pravi (2001, str. 156), da ni jasne ločnice med tem, ali je za slab rezultat kriva zgolj tehnična nezmožnost ali slab management, ampak je v večini primerov mešanica obojega. McManus (2004, str. 97–98) pravi, da bi moral biti glavni namen planiranja managementa tveganj predvsem zagotavljanje najboljšega možnega ravnovesja med zmanjšanjem vplivov tveganj in učinkovitostjo ter stroški razvojnega procesa. Po McManusu je namreč od organizacije odvisno, kolikšno tveganje je pripravljena sprejemati in tudi plačati.

Različne organizacije na različne načine sprejemajo tveganja. Nekatere organizacije so pri poslovanju pripravljene sprejemati večja, druga pa manjša tveganja (angl. *Risk aversion*).

Faza planiranja managementa tveganj mora zato v prvi vrsti podati realno oceno, ali se bo proces managementa tveganj na določenem projektu izplačal ali ne in to na ustrezen način predstaviti naročnikom projekta.

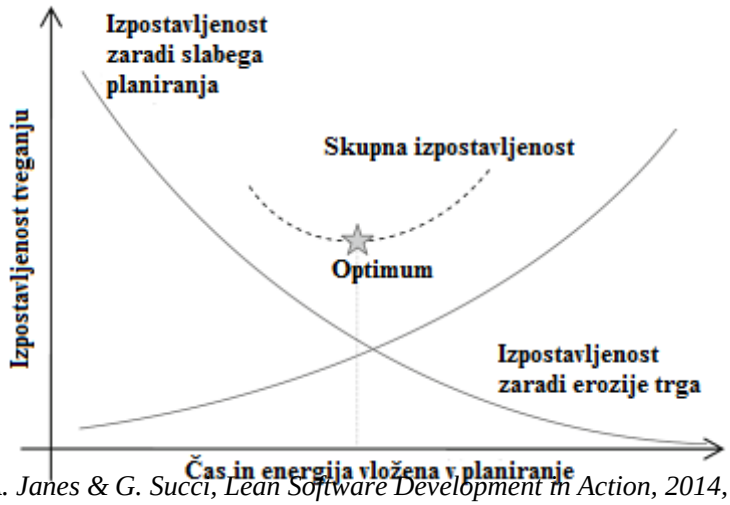
Slika 15: Strateški pomen planiranja managementa tveganj

Vir: H. Kerzner, Strategic Planning for Project Managament using a Project Management Maturity Model, 2011, str. 156.

Janes in Succi (2014, str. 114–116) menita, da je za vsak projekt potrebno oceniti čas in energijo, ki jo bomo vložili v planiranje managementa tveganj tudi zato, ker na izpostavljenost tveganju vplivata tako tveganje, povezano s pomanjkljivim planiranjem managementa tveganj, kot tudi erozija tržnega deleža med trajanjem projekta (tveganje zastarelosti produkta, Slika 16).

V praksi to pomeni, da manj kot vložimo v planiranje managementa tveganj, bolj bomo izpostavljeni tveganjem, vendar bomo hitreje na trgu in s tem zmanjšali tveganje erozije trga ali izgube poslovne priložnosti. Obratno bomo z boljšim planiranjem manj izpostavljeni tveganjem, vendar pa tvegamo, da bo izdelek ob zaključku projekta že zastarel.

Slika 16: Razmerje med izpostavljenostjo tveganju in vložkom v planiranje



Vir: A. Janes & G. Succi, Lean Software Development in Action, 2014, str. 116.

V fazi planiranja managementa tveganj je torej pomembno, da v plan vključimo vse elemente, ki jih navaja PMI (2008, str. 274–278 in 274–278; Tabela 4 in Tabela 5).

Tabela 4: Seznam vhodnih elementov plana managementa tveganj

Št.	Element	Opis
1.	Funkcijske zahteve	Seznam funkcij, ki jih mora izpolnjevati programska oprema.
2.	Stroškovnik	Seznam materialnih stroškov (npr. stroški licenc za razvojna orodja, strojna oprema, merilna oprema, sonde, simulatorji itd.) in stroški izvedbe.
3.	Časovni potek	Grobi potek projekta z razdelitvijo posameznih nalog (angl. *Work breakdown structure*).
4.	Načrt komunikacij	Potek komunikacij med udeleženci projekta in notranjim ter zunanjim okoljem projekta.
5.	Organizacijska sredstva	Arhivske zbirke tveganj iz sorodnih projektov, formati dokumentov in poročil, naloge, pristojnosti in odgovornosti posameznikov v procesu.

Vir: PMI – Project Management Institute, A Guide to the Project Management Body of Knowledge, 2008, str. 274–278.

Tabela 5: Seznam izhodnih elementov plana managementa tveganj

Št.	Element	Opis
1.	Metodologija managementa tveganj	Opis načina, orodij in izvora podatkov, s katerimi bomo izvajali management tveganj na projektu.
2.	Vloge in odgovornosti	Opis vlog posameznikov v procesu managementa tveganj.
3.	Stroškovnik in plan finančnih rezerv	Seznam virov in stroškov, povezanih z managementom tveganj in prikaz oz. razčlenitev finančnih rezerv.
4.	Časovni potek in plan časovnih rezerv	Načrt, kako pogosto bomo izvajali management tveganj in prikaz oz. razčlenitev časovnih rezerv.
5.	Opis kategorij in vrst tveganj	Razčlenitev kategorij in vrst tveganj z opisom (angl. *Risk breakdown structure)*, ki jih lahko pričakujemo na projektu.
6.	Definicija verjetnosti pojava posameznega tveganja in vpliva	Razčlenitev posameznih tveganj po verjetnosti pojava in vplivu na cilje oz. potek projekta.
7.	Definicija območja sprejemljivosti	Definicija območja sprejemljivosti posameznega tveganja in revizija le-teh z naročniki projekta. Namen je, da vnaprej določimo, katero tveganje in v kolikšni meri je na projektu sprejemljivo.
8.	Format poročil	Določitev formata poročil, s katerimi bomo poročali o pojavu tveganj na projektu in poteku aktivnosti za obvladovanje.
9.	Načrt spremljanja tveganj	Načrt, kako in kako pogosto bomo kontrolirali tveganja.

Vir: PMI – Project Management Institute, A Guide to the Project Management Body of Knowledge, 2008, str. 279–280.

2.2.2 Identifikacija tveganj

Po PMI (2008, str. 282) je osnovni cilj faze identifikacije tveganj, da na koncu dobimo čim bolj podroben seznam vseh tveganj (angl. *Risk register*), ki vsebuje naslednje ključne komponente:

1. **seznam identificiranih tveganj**, ki so opisana podrobno, kolikor se le da, vzročnost in posledičnost posameznih tveganj ter njihov vpliv na cilje projekta;
2. **seznam odzivov na tveganja**, ki jih lahko vključimo v plan odzivov na tveganja.

Za uspešno izvedbo procesa identifikacije tveganj potrebujemo poleg elementov, pridobljenih v fazi planiranja managementa tveganj, še:

- **izhodiščno linijo** (angl. *Base line*) ciljev projekta, na podlagi katere bomo ugotavljali odstopanja posameznih ciljev zaradi tveganj na makro in mikro nivoju;
- **plan managementa kakovosti**;
- **projektno dokumentacijo**;
- **ocene zunanjega in notranjega okolja projekta**;
- **spoznanja iz prejšnjih projektov** (angl. *Lessons learned).*

Za identifikacijo tveganj se lahko poslužujemo različnih metod, kot so npr.:

- **ekspertne analize**;
- **nevihta možganov** (angl. *Brainstorming*);
- **analiza PSPN** (a*naliza prednosti, slabosti, priložnosti in nevarnosti,* angl. *SWOT);*
- **analiza ključnih izvorov in učinkov tveganj** (angl. *Root cause effect analysis*);
- **analize vzrokov in učinkov** (angl. *Failure mode and effect analysis – FMEA*) itd.

V procesu identifikacije tveganj naletimo vedno znova na nova tveganja, ki jih v obliki „lekcij iz preteklosti“ (angl. *Lessons learned*) vključimo v seznam (oz. register, angl. *Risk register*) tveganj. Na ta način sčasoma ustvarimo obširen seznam oz. bazo tveganj (angl. *Risk base*), ki nam pomaga pri učinkovitejši identifikaciji tveganj.

2.2.3 Kvalitativna analiza tveganj

Kot smo omenili prej, v praksi ne moremo planirati odzivov na vsa tveganja, ki smo jih identificirali v procesu identifikacije, oz. bi to zahtevalo preveč časa in virov. Zato je namen kvalitativne analize tveganj, da postopek racionaliziramo in identificirana tveganja ovrednotimo ter se s tem osredotočimo le na tista, ki se bodo najverjetneje pojavila in najmočneje vplivala na potek in rezultat projekta (PMI, 2008, str. 289).

Pri tem je pomembno, da poskušamo v tem procesu odpraviti vpliv pristranskosti, saj udeleženci v procesu identifikacije tveganj tveganja identificirajo na podlagi lastnih izkušenj ali nagibov. Mulcahy-jeva (2010, str. 131) navaja dve ključni vrsti pristranskosti in sicer: motivacijsko in kognitivno pristranskost.

2.2.3.1 Motivacijska pristranskost

Pri motivacijski pristranskosti gre predvsem za zavedno pristranskost, ki temelji na osebnem dojemanju situacije in želji po uveljavitvi ter vplivu na potek projekta.

Ritchie-Dunham (2013, str. 4) je mnenja, da je glavna motivacijska osnova za pristranskost osebno dojemanje nagrade in kazni. Tako bodo udeleženci v procesu identifikacij tveganj želeli biti nagrajeni na lastnem področju in bodo težili k temu, da bodo kot visoko tvegane ocenjevali konkurenčna področja ali izvajalce, ne pa tudi lastnih.

Nasprotno udeleženci identifikacijskega procesa ne bodo želeli biti kaznovani za slab končni rezultat projekta, zato bodo poskušali neznana področja oceniti kot visoko tvegana in se s tem zavarovati pred odgovornostjo za slab rezultat.

Po izkušnjah iz prakse bodo udeleženci identifikacijskega procesa poskušali, zaradi časovnega tveganja in bojazni po nepravočasni izvedbi del, zmanjševati dogovorjeni funkcijski obseg s tem, da bodo implementacijo določenih funkcij ovrednotili kot zelo tvegano. Prav tako je pri razvijalcih programske opreme zelo pogosta nagnjenost k „problemom“ in manj k „rešitvam“.

Temu se manager projekta lahko izogne tako, da si najprej sam ustvari zelo dobro sliko o želenih ciljih in končnem rezultatu projekta in to sliko čim bolj natančno posreduje razvijalcem, oz. da ves čas projekta skrbi, da je slika končnega rezultata projekta med razvijalci in naročniki čim bolj usklajena.

2.2.3.2 Kognitivna pristranskost

Pri kognitivni pristranskosti gre predvsem za nezavedno razliko v dojemanju in procesiranju informacij.

Ritchie-Dunham (2013, str. 4) navaja, da je osnovna težava v tem, da:

- so ocene, ki jih niso podali eksperti, zelo slabe in imajo veliko varianco;
- eksperti dajejo dobre ocene, z majhno varianco, ampak so v svojih ocenah preveč optimistični, ker izhajajo iz lastnih sposobnosti in izkušenj.

Kognitivno pristranskost, kot navaja Ritchie-Dunham (2013, str. 8–18), najlažje obvladujemo tako, da v začetku podamo osnovno oceno tveganja, ki jo skozi iterativni proces izpopolnjujemo. S tem ne-ekspertom omogočimo, da izgubijo strah in svoje ocene podkrepijo s podatki in temeljiteje raziščejo, ekspertom pa preprečimo, da se postavijo na stališče, ki ga hočejo s svojo ekspertizo na vsak način braniti.

2.2.3.3 Kvalitativno ocenjevanje projektnih tveganj

Za kvalitativno oceno posameznih projektnih tveganj je v prvi vrsti pomembno, da podamo oceno verjetnosti pojava tveganja (**p**) (angl. *Probability*) in vpliv na rezultat projekta (**i**) (angl. *Impact*). Mulcahy-jeva (2010, str. 132) predlaga, da se za zmanjšanje pristranskosti pri zbiranju kvalitativnih ocen posameznih tveganj poslužujemo vnaprej določenih pomenov ocen za verjetnost pojava in vpliv tveganja od 1 do 10 (Tabela 6 in Tabela 7).

Tabela 6: Primer ocenjevanja stopnje verjetnosti pojava tveganja (p)

Ocena	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Verjetnost	Nizka		Srednja		Nizko-Visoka		Visoka		Dejstvo	

Vir: R. Mulcahy, Risk Management, Tricks of the Trade for Project Management, 2010, str. 131.

Tabela 7: Primer predpisa za oceno vpliva tveganja (i)

Ocena	**Definicija vpliva**
10	Neuspeh projekta
9	Prekoračitev stroškov ali časa izvedbe za več kot 40 %
8	Prekoračitev stroškov ali časa izvedbe za 30–40 %
7	Prekoračitev stroškov ali časa izvedbe 20–30 %
6	Prekoračitev stroškov ali časa izvedbe 10–20 %
5	Prekoračitev stroškov ali časa izvedbe < 10 %
4	Velik vpliv na zmanjšanje finančnih in časovnih rezerv projekta

3	Srednji vpliv na zmanjšanje finančnih in časovnih rezerv projekta
2	Majhen vpliv na zmanjšanje finančnih in časovnih rezerv projekta
1	Ne vpliva

Vir: R. Mulcahy, Risk Management, Tricks of the Trade for Project Management, 2010, str. 132.

Glede na prejšnje ugotovitve v zvezi z standardnim odstopanjem projektov glede na raven organizacijske zrelosti, bi bilo smiselno vrednosti prekoračitve stroškov ali časa izvedbe razširiti, saj so vrednosti, ki jih navaja Mulcahy-jeva, s stališča razvoja programske opreme precej konservativne. Realnejšo sliko prikazujeta DeMarco in Lister (2003, str. 104–111) in je zato ustreznejša spremenjena ocena vpliva tveganja (Tabela 8).

Tabela 8: Popravljen primer predpisa za oceno vpliva tveganja (i)

Ocena	**Definicija vpliva**
10	Neuspeh projekta
9	Prekoračitev stroškov ali časa izvedbe za več kot 200 %
8	Prekoračitev stroškov ali časa izvedbe za 175 %
7	Prekoračitev stroškov ali časa izvedbe za 150 %
6	Prekoračitev stroškov ali časa izvedbe za 125 %

se nadaljuje

nadaljevanje

5	Prekoračitev stroškov ali časa izvedbe za 100 %
4	Prekoračitev stroškov ali časa izvedbe za 75 %
3	Prekoračitev stroškov ali časa izvedbe za 50 %
2	Standardna prekoračitev stroškov ali časa izvedbe za 30 %
1	Ne vpliva

2.2.3.4 Kvalitativno ocenjevanje tehničnih tveganj

Pri razvoju vgrajene programske opreme **je smiselno, ni pa nujno**, pri tehničnih tveganjih vključiti v oceno vpliva tveganja tudi „ranljivost“ (*v*) (angl. *Vulnerability*).

Pri tem ne gre zgolj za ranljivost s stališča varnosti pred zunanjimi vdori in manipulacijo programske opreme ali kraje osebnih podatkov, ampak tudi za oceno vplivov, ki izhajajo iz načina izvedbe (implementacije) in uporabe programske opreme.

Dodajanje tretjega (opcijskega) parametra v oceno posameznega tehničnega tveganja je smiselno, kadar se s sicer visoko ocenjenimi tveganji ne ukvarjamo (v primeru, ko je vpliv na

končen izdelek neopazen), ampak se osredotočimo na nižje ocenjena tveganja z visokim vplivom.

Za oceno lahko uporabimo podobno tabelo kot za oceno verjetnosti pojava tveganja (Tabela 6), v kateri podamo, kakšno (morebitno) ranljivost bo določena rešitev predstavljala za končno uporabnost izdelka (Tabela 9).

Tabela 9: Primer ocenjevanja stopnje ranljivosti (v)

Ocena	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Ranljivost	Neopazno		Zanemarljivo		Pogoste motnje		Delno neuporabno		Neuporabno	

S stališča ocenjevanja tehničnih tveganj pri vgrajeni programski opremi lahko posamezne stopnje ranljivosti interpretiramo s pomočjo klasifikacije motenj, ki se pojavljajo pri obratovanju (Tabela 10).

Tabela 10: Klasifikacija motenj za oceno ranljivosti

Razvrstitev	**Opis**
Neopazno	Neopazne motnje so motnje v delovanju, ki jih uporabnik ne bo opazil. Pri implementaciji razumemo to kot „običajne“ težave postranskega pomena.
Zanemarljivo	So občasne motnje v delovanju, ki jih eventualno opazimo, če smo zelo pozorni, sicer pa ne. To razumemo kot manjšo težavo, ki jo enostavno odpravimo in za proces razvoja ne predstavlja večje motnje.
Pogoste motnje	So „tihe“ motnje pri delovanju, ki so še vedno opazne, ampak le, če smo dovolj pozorni in ne vplivajo kritično na delovanje sistema (angl. *Fail silently*). To so motnje, ki zahtevajo prilagoditve, dodatna preizkušanja ali poglobljeno analizo napak.
Delno neuporabno	So pogoste motnje, ki predstavljajo težave pri uporabi, vendar je izdelek ob nastopu motnje še vedno uporaben (angl. *Fail operational*). To pomeni, da določeni programski sklopi sicer delujejo, ampak povzročajo veliko težav in zahtevajo veliko virov pri analizi in odpravi napak.
Neuporabno	Izdelek ob pojavu motnje postane nefunkcionalen in ga ne moremo uporabljati. To pomeni, da je določena rešitev povsem neuporabna in je tudi s prilagoditvami ni mogoče uspešno uporabiti brez ponovne zasnove in izvedbe.

2.2.3.5 Skupna ocena verjetnosti tveganja

Na podlagi zbranih ocen posameznikov v procesu ocenjevanja identificiranih tveganj izračunamo zmnožke verjetnosti (p) in vpliva (i) ter opcijsko ranljivost (v) za vse ocene posameznega tveganja (R_n), ki so smo jih pridobili v postopku ocenjevanja:

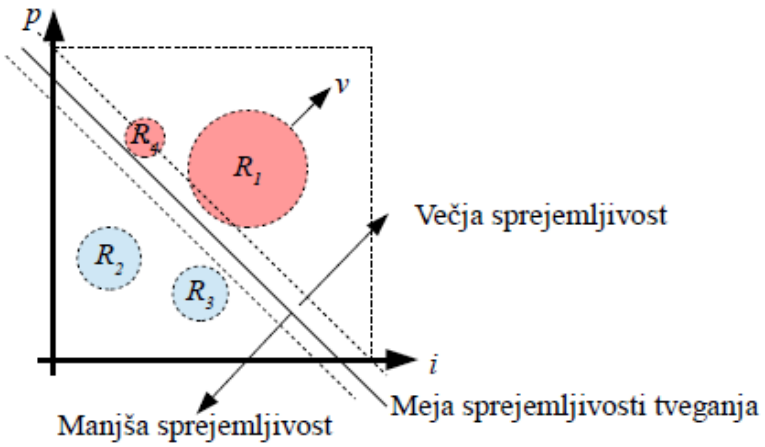
$$R_n = p_n \times i_n \times (v_n) \tag{1}$$

in izračunamo mediano vseh zmnožkov:

$$\widetilde{R} = [R_1 \ldots R_n] \tag{2}$$

Tako izračunana vrednost predstavlja ocenjeno vrednost vpliva posameznega tveganja (R). Pri nadaljevanju procesa identificirana in ocenjena tveganja razvrstimo po njihovem vplivu od najvišje do najnižje ocenjene (angl. *Risk ranking*). V skladu z nivojem, določenim s sprejemljivostjo projektnega tveganja (angl. *Risk acceptance threshold*), upoštevamo oz. se v nadaljevanju aktivno ukvarjamo le s tveganji, katerih ocena leži izven ali na meji sprejemljivega območja tveganja (Slika 17). S tveganji, ki ležijo znotraj sprejemljivostnega območja, se ne ukvarjamo aktivno, je pa pomembno, da jih v procesu kontrole tveganj vedno znova preverimo in ocenimo njihov vpliv glede na trenutno stanje projekta. Pri tem je potrebno tudi upoštevati, da se meja sprejemljivosti tveganja v času projekta zaradi notranjih ali zunanjih vplivov na projekt lahko spreminja.

Slika 17: Meje sprejemljivosti tveganja



2.2.4 Kvantitativna analiza tveganj

Po definiciji PMI (2008, str. 294) je kvantitativna analiza tveganj numerična analiza identificiranih in po prioriteti razvrščenih tveganj, z namenom kvantitativne ocene vpliva tveganj na rezultat projekta. Ta korak v procesu managementa tveganj ni nujno potreben, ker zahteva veliko časa in poglobljeno analizo. Zato je vire, potrebne za tovrstno analizo, bolj smiselno nameniti procesu identifikacije in kvalitativne ocene tveganj (Mulcahy, 2010, str. 164). Bistveni elementi kvantitativne analize tveganj so:

- **ocena časovnega razpona** (vpliv na časovni potek projekta);
- **ocena finančnega razpona** (vpliv na stroške projekta);
- **ocena kvalitativnega razpona** (vpliv na kakovost rezultata projekta);
- **ocena vsebinskega razpona** (vpliv na vsebino oz. izpolnitev projektnih zahtev).

V literaturi najdemo različne metode in pristope h kvantitativni analizi. Avtorji opisujejo kot najbolj primerne naslednje metode:

- metoda pričakovanih denarnih vrednosti (angl. *EMV – Expected monetary value*);
- metoda drevesa odločanja (angl. *Decision tree analysis*);
- analiza Monte Carlo;
- izračun neto sedanje vrednosti (angl. *NPV – Net present value*);
- izračun interne stopnje donosa (angl. *IRR – Internal rate of return*);
- analiza diskontiranih denarnih tokov (angl. *DCF – Discounted cash flow*);
- analiza stroškov in koristi (angl. *Cost benefit analysis*).

Pri projektih razvoja vgrajene programske opreme je, poleg metode pričakovanih denarnih vrednosti (v nadaljevanju EMV), najbolj uporabna metoda drevesa odločanja, pri kateri lahko na podlagi seznama tveganj in ocen verjetnosti že v fazi načrtovanja projekta sprejmemo odločitve v zvezi z:

- nakupom ali lastno izvedbo modulov (angl. *Make or buy*);
- vključevanjem zunanjih izvajalcev in ceno njihovega vključevanja (angl. *Outsourcing cost*);
- definicijo funkcijskega oz. kvalitativnega obsega (angl. *Function/quality scope*).

2.2.5 Planiranje odzivov na tveganja in priložnosti

Po definiciji PMI (2008, str. 301) je planiranje odzivov na tveganja proces razvoja opcij in aktivnosti, ki med potekom projekta pomagajo povečati priložnosti in zmanjšajo pojav in vpliv tveganj na rezultat projekta. DeMarco in Lister (2003, str. 71) poudarjata, da je bistven element za učinkovito kontrolo pojava tveganj natančno definiranje indikatorjev materializacije tveganj in odzivov že v fazi planiranja. Pri tem obravnavamo posamezna tveganja po njihovi prioriteti, določimo kriterije in kontrolne točke (angl. *Triggering condition and threshold*). Za aktivnosti, povezane z odzivom na tveganje, določimo odgovorno osebo in rezerviramo vire, ki jih bomo uporabili za odpravo tveganja in njegovih posledic.

2.2.5.1 Odzivi na tveganja

V literaturi (npr. PMI, 2008, str. 303–304; Mulcahy, 2010, str. 195) najdemo naslednje strategije odzivov na tveganja:

1. **Izogib tveganju** (angl. *Avoidance*) – odprava ključnega razloga, ki bi imel za posledico uresničitev tveganja. Metode, ki jih lahko uporabimo, so: podaljšanje trajanja projekta, zmanjšanje funkcijskega obsega ali zavestno zmanjšanje kakovosti.
2. **Zmanjšanje vpliva tveganja** (angl. *Mitigation/control*) – zmanjšanje pričakovane denarne vrednosti tveganja z zmanjšanjem vpliva ali verjetnosti pojava tveganja. Metode, ki jih lahko uporabimo za zmanjšanje vpliva tveganja, so: povečanje kontrolnih točk ali vmesnih ciljev projekta, poenostavitev potekov posameznih faz, razdelitev projekta na zaključene pod-projekte, prototipiranje, vgraditev dodatnih kontrolnih ali redundantnih mehanizmov, ki zmanjšujejo pojav sekundarnih tveganj.
3. **Prenos tveganja** (angl. *Transfering/allocation*) – prenos tveganja na drug subjekt, npr. na zavarovalnico ali podizvajalca. Pri tem moramo upoštevati, da prenos tveganja le-tega ne odpravlja, ampak ga samo prenaša na drug subjekt (proti plačilu). Tako moramo v tem primeru upoštevati povišanje stroškov projekta.
4. **Sprejemanje tveganja** (angl. *Acceptance*) – pasivno sprejemanje tveganja po načelu „če se bo zgodilo, se bo pač zgodilo“. Kljub temu teh tveganj ne smemo zanemariti in jih moramo vnaprej dokumentirati, kako bomo ravnali v primeru nastanka.

Mulcahy-jeva (2010, str. 196) poudarja, da ne glede na to, katero strategijo odziva bomo za posamezno tveganje izbrali, moramo upoštevati da:

- mora biti odziv pravočasen;
- obseg in cena potrebnih aktivnosti za preprečitev uresničitve tveganja mora biti manjša od pričakovane denarne vrednosti (EMV) vpliva tveganja na projekt;
- posamezen odziv ni izolirana dejavnost, ampak lahko pokriva več tveganj hkrati.

2.2.5.2 Odzivi na priložnosti

PMI (2008, str. 303–304) priporoča naslednje strategije odzivov na priložnosti:

- **izraba priložnosti** (angl. *Exploitation*) – povečanje pozitivnega vpliva priložnosti na potek projekta z dodelitvijo virov, ki zagotavljajo uspeh (npr. z vključevanjem najbolj talentiranih razvijalcev ali investiranjem v zmogljivejšo razvojno opremo);
- **delitev priložnosti** (angl. *Sharing*) – povečevanje pozitivnega vpliva z vključevanjem posebnih timov ali zunanjih izvajalcev s posebnim ekspertnim znanjem ali tehnologijo;
- **povečanje** (angl. *Enhancing*) – strategija, ki jo uporabljamo za ugotovitev ključnega dejavnika, ki lahko zagotovi pozitiven vpliv na potek projekta;
- **sprejemanje** (angl. *Acceptance*) – pasivno sprejemanje pozitivnega vpliva priložnosti.

2.2.6 Planiranje rezerv

Praksa kaže, da se managerji pri procesu načrtovanja projektov največkrat zatekajo k planiranju rezerv („čez palec“) in sicer na podlagi izkušenj iz prejšnjih projektov ali priporočil iz prakse. Takšen pristop je slab že v osnovi, saj je po definiciji vsak projekt enkraten podjem, torej ne moremo sklepati, da bodo zanj veljale enake zakonitosti. Ni potrebno posebej poudarjati, da projekti s takšnim pristopom k načrtovanju rezerv brez potrebe stanejo bistveno več, kot je potrebno.

Pri razvoju vgrajene programske opreme velja praktično priporočilo (glej pog. 1.2), da naj bi projektna rezerva znašala najmanj 30 %, čeprav so anketirani managerji povedali, kot navaja Dua (2012, str. 6), da bi ta rezerva pri večini projektov lahko brez težav znašala 100–200 %. Dua (2012, str. 3) prav tako opozarja, da gre po eni strani za pomanjkljivo razumevanje vrst in pomena rezerv (oz. tveganj), po drugi strani pa za slabo komunikacijo nasproti naročnikom projekta.

Tako Dua kot Mulcahy-jeva (2010, str. 210–217) omenjata dve vrsti rezerv, ki bi jih morali upoštevati pri načrtovanju :

- **projektne (managerske) rezerve** (angl. *Management reserves*);
- **varnostne rezerve** (angl. *Contingency reserves*).

PMI (2008, str. 72, 151, 156, 158 itd.) varnostne rezerve povezuje predvsem z rezervami v časovnem poteku projekta (angl. *Scheduling reserves*).

S stališča razvoja programske opreme je to pomanjkljivo (glej pog. 1.2), saj gre za projektne rezerve, ki jih moramo tako ali tako upoštevati pri načrtovanju projekta, ker ni mogoče vnaprej predvideti potrebnega obsega del (vrstic programske kode). Torej gre za rezerve, ki so potrebne zaradi same narave projekta in v katerih dejanska (tehnična in ostala) tveganja specifičnega projekta sploh niso upoštevana.

Pomembno razliko med projektnimi in varnostnimi rezervami opisuje Dua (2012, str. 4–5). Pravi, da so projektne rezerve del samega projekta in so v „lasti“ (angl. *Ownership*) projektnega managerja. To so v bistvu časovne in denarne rezerve, ki predstavljajo „tampon“ (angl. *Buffer*) za pokrivanje negotovosti (angl. *Uncertainty*) v fazi načrtovanja projekta.

Dua (2012, str. 4–5) nadalje pravi, da so varnostne rezerve „izven projektne“ rezerve, ki niso v lasti projektnega managerja in niso nekaj, česar bi se posluževali v primeru težav na projektu. Uporaba varnostnih rezerv mora biti namenjena izključno za odpravljanje posledic ob materializaciji tveganj ali za preprečevanje le-teh.

Dua (2012, str. 5–6) opozarja tudi na „10 % sindrom“, kadar kot varnostno rezervo preprosto dodamo 10 % v času in denarju. Dua je mnenja, da je to dokazano napačen pristop in navaja priporočila ameriške Agencije za energetiko (angl. *Department of Energy*), ki priporoča, da

naj začetne varnostne rezerve obsegajo 30–70 % predvidenih (ocenjenih) stroškov za dokončanje projekta, v nadaljevanju ETC (angl. *Estimated cost to complete*), ki se po podrobni analizi tveganj ali tekom projekta lahko zmanjšajo na 15–55 % ETC. Mulcahy-jeva (2010, str. 210) pravi, da je planirano trajanje projekta (t) odvisno od trajanja kritične poti (t_{KP}), ki mu prištejemo še planirano projektno rezervo (t_{PR}) in varnostno rezervo (t_{VR}).

$$t = t_{KP} + t_{PR} + t_{VR} \tag{3}$$

Enako velja za stroške projekta (C), torej da načrtovanim stroškom izvedbe projekta (C_P) prištejemo projektno rezervo (C_{PR}) in varnostno rezervo, ki jo izračunamo na podlagi vsote ocenjenih denarnih vrednosti (C_{EMV}) posameznih tveganj.

$$C = C_P + C_{PR} + \sum_{i=1}^{n} C_{EMV} \tag{4}$$

Tako Dua kot Mulcahy-jeva priporočata uporabo kvantitativnih metod za oceno obsega varnostne rezerve, saj nam le-ta omogoča, da jo načrtujemo bolj racionalno in jo pred naročniki projekta tudi lažje utemeljimo.

2.2.7 Kontroliranje tveganj

Osnovni namen kontroliranja tveganj je odprava ali zmanjšanje pojava in vpliva tveganj na sprejemljiv nivo znotraj območja sprejemljivosti. Po PMI (2008, str. 308) je kontroliranje in obvladovanje tveganj nenehen proces, ki ga izvajamo tekom projekta in katerega namen je:

- kontroliranje, ali projektna izhodišča še vedno veljajo;
- kontroliranje, ali so identificirana tveganja in planirani odzivi na tveganja še vedno aktualni (tj. ocena sprememb dejavnikov tveganja);
- kontroliranje tveganj po prioriteti in oceni posameznih tveganj (npr. sprememba ocene vpliva posameznega tveganja);
- identifikacija novih tveganj, njihova obravnava, prilagoditev poteka projekta in ocena vpliva na količino potrebnih rezerv;
- kontroliranje stanja projektnih rezerv;
- kontrola, ali se management tveganj dejansko izvaja tako, kot je bil načrtovan;
- kontrola učinkovitosti izpeljanih ukrepov za zmanjšanje tveganj.

Za kontrolo tveganj je torej nujno potrebno, da vnaprej definirano, kako bomo merili pojav in vpliv tveganj. Kot smo povedali uvodoma, je tudi potrebno, da definiramo območje sprejemljivosti posameznega tveganja. Proces kontroliranja tveganj poteka tako, da je kontroliranje posameznih tveganj razdeljeno med lastnike tveganj, katerih naloga je, da čim hitreje odkrijejo simptom uresničevanja tveganja in pravočasno izvedejo morebitne planirane korektivne ukrepe (Stare 2011c, str. 264). Manager projekta na rednih kontrolnih sestankih preverja stanje tveganj in po potrebi dopolnjuje seznam. Pogostost sestankov je odvisna od verjetnosti uresničitve in vpliva tveganja na projekt. Večje kot je tveganje, pogosteje ga je treba preverjati (Stare 2011c, str. 264).

3 ANALIZA PROJEKTA X

Projekt X je tipičen „ad hoc“ razvojni projekt programske opreme, ki je nastal zaradi potrebe po prikazu parametrov vozila v realnem času, torej med preizkusno vožnjo in sicer na mobilnih (prenosnih) napravah, kot so npr. Apple iPad ali podobne tablice z operacijskim sistemom Android, Linux ali Windows. Ker mobilne naprave omogočajo brskanje po spletu, se je hitro porodila ideja, da bi merilna naprava oblikovala spletno stran in jo posredovala na spletni brskalnik mobilne naprave, kjer bi s pomočjo Java aplikacije potekal prikaz merilnih signalov.

Kmalu se je izkazalo, da tovrstna aplikacija, zaradi omejenih zmogljivosti mobilnih naprav, ni dovolj hitra in da je za opazovanje merilnih veličin v realnem času praktično neuporabna. Na podlagi ugotovitev se je pričel projekt razvoja mobilne aplikacije, ki bi lahko kot namenska aplikacija tekla na mobilnih napravah (z operacijskimi sistemi Android, iOS, Linux in Windows).

Ker mobilne naprave večinoma ne omogočajo žične povezave z drugimi napravami, je bilo potrebno, poleg razvoja programske opreme, razviti še vmesnik za brezžični prenos podatkov (WLAN) med merilno in mobilno napravo. Da bi bil prikaz podatkov za uporabnike čim bolj prilagodljiv, je bilo potrebno izdelati aplikacijo, ki omogoča konfiguracijo instrumentov na PC računalniku. Konfiguracija instrumentov poteka s pomočjo baz merilnih signalov v vozilu in mora biti poleg tega zaščitena (šifrirana) za primer kraje testnega vozila in/ali tablice.

Potek projekta X je obsegal izvedbo naslednjih projektnih nalog:

- razvoj WLAN komunikacijskega vmesnika za brezžični prenos podatkov;
- razvoj gonilnika za komunikacijski vmesnik v merilni napravi;
- razvoj vgrajenega strežnika na merilni napravi, ki iz podatkov v vozilu v realnem času izlušči podatke in jih nato v šifrirani obliki posreduje na mobilno napravo po brezžični povezavi;
- razvoj programske opreme (App), ki teče na različnih mobilnih napravah;

- razvoj programske opreme za konfiguracijo na PC računalniku.

3.1 Potek projekta X

V nadaljevanju bomo skicirali potek projekta, kaj je bilo izvedeno v določenih fazah projekta in kje so nastopile težave:

1. **Prva (prototipna) faza**:

- razvoj spletne aplikacije;
- uporaba komercialno dostopnega WLAN modula;
- standardni grafični instrumenti za prikaz informacij.

2. **Druga faza**:

- aplikacija za Windows in Linux operacijski sistem;
- slabo delujoče verzije za Android in iOS operacijski sistem;
- prvi WLAN vmesnik brez dostopne točke.

3. **Tretja faza**

- delujoča aplikacija za mobilne operacijske sisteme Android in iOS;
- standardni grafični instrumenti za prikaz podatkov;
- WLAN vmesnik s funkcijo dostopne točke;
- slaba kakovost aplikacije, veliko majhnih napak;
- aplikacija pri večji obremenitvi na počasnejših tablicah ni bila stabilna.

4. **Četrta faza**

- obširna razširitev nabora funkcij na zahtevo uporabnikov;
- konfigurabilni instrumenti za prikaz podatkov;
- izboljšanje konfiguracijskega programa;
- optimizacija zmogljivosti in zanesljivosti delovanja.

3.1.1 Težave na projektu X

V nadaljevanju bomo opisali težave, ki so nastopile v posameznih fazah projekta. Težav v prvi, prototipni fazi ne bomo obravnavali, saj je šlo za t. i. preizkus izvedljivosti ideje (angl. *Proof of concept, Feasibility study*).

Osnovna težava je bila v tem, da so se ves čas projekta spreminjale oz. dopolnjevale osnovne zahteve projekta, saj so s strani razvijalcev, kakor tudi potencialnih kupcev, prihajale vedno nove ideje oz. predlogi za izboljšave.

3.1.1.1 Težave v drugi fazi projekta

Prva težava je bila iskanje WLAN vmesniškega modula, ki bi s centralno procesno enoto lahko komuniciral po USB vmesniku in bi hkrati deloval znotraj zahtevanega temperaturnega območja od -40ºC do 85ºC. S tem bi se podjetje namreč izognilo modifikacijam strojne opreme in prihranilo pri stroških projekta.

Nadalje bi moral biti gonilnik za modul podprt na operacijskem sistemu Linux, na katerem teče merilna naprava. Kmalu se je izkazalo, da je na trgu primernih modulov na pretek, vendar so po večini specificirani za komercialno temperaturno območje od 0ºC do 60ºC ali še slabše in so torej neuporabni.

Nadalje je bilo potrebno razmisliti o poenotenju programske opreme, ki bo tekla na mobilnih napravah, saj nima smisla razvijati programske opreme za vsak operacijski sistem posebej. Po raziskavi komercialno dostopnih rešitev na trgu je podjetje kmalu našlo programsko knjižnico, ki je obljubljala enostavno rešitev problema, hkrati pa je bila dostopna tudi v odprto kodni obliki, s čimer se je podjetje želelo zavarovati pred tveganjem nakupa neustrezne programske knjižnice.

Ker je bila knjižnica ob začetku projekta za mobilno platformo iOS na voljo šele v testni različici, je bilo tveganje, da se bodo na tej platformi pojavljale težave pri realizaciji, dokaj visoko, kar se je pozneje tudi uresničilo.

Največji udarec za projekt je nastopil, ko je bil izdelan vmesnik za brezžično komunikacijo. Razvijalci so popolnoma spregledali dejstvo, da mora takšen vmesnik predstavljati tudi dostopno točko za mobilne naprave, sicer se mobilne naprave z vmesnikom po brezžični povezavi ne morejo povezati. Uporabljeni modul te opcije ni omogočal oz. jo je omogočal le z velikimi omejitvami, kar pa v danem primeru ni bila uporabna rešitev.

3.1.1.2 Težave v tretji fazi projekta

Zaradi pomanjkanja virov, predvsem pa zaradi odpovedi ključnega razvijalca, se je podjetje odločilo, da bo za večji del izvedbenih nalog tretje faze najelo zunanjega izvajalca, ki je že imel izkušnje pri razvoju aplikacij za operacijska sistema Android in iOS. Cena storitev zunanjega izvajalca je bila zelo atraktivna, zato management ni dolgo odlašal z oddajo del.

Pri predaji del in planiranju aktivnosti se je pri zunanjem izvajalcu kmalu po začetku sodelovanja izkazalo, da je specifikacija programske opreme s stališča izvajalca premalo podrobna in da razvijalci niso popolnoma razumeli, kakšen točno naj bil izdelek oz. končni cilj projekta.

Prva delujoča preizkusna verzija je bila s strani zunanjega dobavitelja dokončana v rekordnem času. Po validaciji izdelka in reviziji zahtev pa je prišlo do prvega konflikta. V zahtevah je bilo navedeno tudi, da morajo biti grafične komponente, ki prikazujejo merjene vrednosti, prosto nastavljive na PC računalniku in sicer po principu «tako kot se vidi na PC računalniku, tako se mora videti tudi na ciljni napravi (angl. *What you see is what you get*).

To zahtevo je zunanji dobavitelj popolnoma spregledal, kar je pomenilo, da za končni izdelek ne more več uporabiti standardnih prikazovalnikov, ki so bili na voljo v uporabljeni grafični knjižnici, ampak mora izdelati nove. V tem trenutku je prišlo do streznitve tudi na strani zunanjega izvajalca, saj je postalo jasno, da so podcenili obseg del in da projektne zahteve niso trivialne.

Veliko težav je nastopalo tudi pri poenotenju grafičnih komponent, razvitih v programskem jeziku C++ in njihovi vključitvi v aplikacijo za konfiguriranje, ki je bila napisana v programskem jeziku C#. Zato je bilo potrebno najti rešitev, kako module, napisane v različnih programskih jezikih, združiti v uporabno celoto in revidirati projekt.

Po zaključku tretje faze je bilo ugotovljeno, da je:

- projekt v prvih treh fazah prvotno načrtovane stroške presegel za že več kot 170 %;
- bila tretja faza zaključena 4 mesece prepozno;
- bil funkcijski obseg po zaključku tretje faze približno 75 % prvotnih projektnih zahtev.

Kljub dokaj slabemu razpoloženju naročnikov projekta je bila, zaradi velikega interesa ključnih kupcev po nadgrajeni verziji, sprejeta odločitev, da dobi projekt finančno podporo še za eno fazo. Naročniki projekta so obenem zahtevali tudi natančni finančni in časovni plan projekta in seznam tveganj ter posledic za potek projekta. Zato smo se odločili, da se bomo managementa tveganj na projektu lotili bolj sistematsko, da bi s tem zagotovil uspešen zaključek projekta.

3.2 Management tveganj na projektu X

Očitno je, da na projektu X v začetnih treh fazah ni bilo ustreznega procesa managementa tveganj, kar je imelo za posledico nezadovoljstvo naročnikov projekta zaradi prekoračitve stroškov, poznega lansiranja prve verzije aplikacije in navsezadnje tudi kupcev aplikacije, ki s kakovostjo prve uradne različice niso bili zadovoljni.

Če pogledamo nazaj na sam potek projekta, lahko rečemo, da je projekt potekal v skladu z inkrementalnim razvojnim modelom, saj smo prvo uradno različico aplikacije razvili v treh fazah. Najprej smo razvili prototip, nato dve neuradni različici, vsakič z večjim funkcijskim obsegom in na koncu prvo uradno različico. Zaključimo lahko, da je bil izbran razvojni proces ustrezen in da na tem projektu ni bil vir tveganj. Kljub temu se je v vsaki fazi zatikalo zaradi tehničnih težav, kar je imelo za posledico, da ni bil izveden celotni seznam funkcijskih zahtev. V vsaki fazi projekta, z izjemo prototipne, smo večji del časa porabili tudi za popravljanje napak iz predhodne faze.

Iz opisanih težav lahko sklepamo, da je šlo pri tem projektu predvsem za težave zaradi neustreznega managementa in procesnih napak, manj pa za težave zaradi tehnične (v smislu pomena managementa tveganj (Slika 15). Pri tem gre neustreznemu managementu pripisati predvsem to, da smo se premalo ukvarjali s podrobnejšo analizo, predvsem tehničnih tveganj in procesom obvladovanja tveganj.

Čeprav je bilo jasno, da se bodo na projektu pojavljala predvsem tehnična tveganja (npr. uporaba kupljene grafične knjižnice, nove mobilne platforme, dovolj hiter prenos podatkov, prekinitve v prenosu podatkov, uporaba iste kode na različnih platformah), nismo ustrezno ocenili vpliva teh tveganj na projekt, kot tudi ne predvideli odzivov na tveganja.

3.2.1 Planiranje managementa tveganj

Na tem projektu procesa planiranja managementa tveganj, kot smo ga opisali v poglavju 2.2.1, v začetnih fazah ni bilo. Zato smo se pred začetkom četrte faze odločili, da bomo v projektni plan četrte faze vključili tudi plan managementa tveganj in da bomo za planiranje namenili 15% finančnih virov projekta. V ta namen je bila izdelan plan managementa tveganj v skladu s priporočili PMI (Tabela 4 in Tabela 5).

3.2.2 Splošna (hitra) ocena tveganosti projekta

Na projektu X smo najprej izvedli hitro oceno tveganosti in sicer tako, da smo sestavili preglednico na podlagi tabele najpogostejših virov tveganj (Tabela 3), v kateri smo ocenili stopnjo pokritja za vse vire najpogostejših tveganj s seznama.

Stopnja pokritja nam pove, v kolikšni meri smo zagotovili vse potrebno za izločitev posameznih možnih virov tveganja. Na ta način dobimo »radarsko sliko« virov tveganj po posameznih področjih, kar po našem mnenju lahko pomaga pri identifikaciji projektnih tveganj, saj se na ta način osredotočimo na področja z večjo izpostavljenostjo.

S tem pridobimo na času, saj na takšen način zelo hitro ocenimo, v kolikšni meri je projekt tvegan, na katerih področjih lahko pričakujemo težave in ali se splača vložiti dodatne vire v podrobnejšo identifikacijo (angl. *Quick risk assesment test*).

Postopek je potekal tako, da je bil najprej ocenjen prispevek k skupnemu tveganju posameznega področja (Priloga 1). Posamezne vire tveganj posameznega področja (D) smo enakomerno porazdelili (utežna vrednost) tako, da skupaj predstavljajo 100 % posameznega področja. Utežno vrednost D smo določili na podlagi preteklih izkušenj.

Za vsak vir tveganja smo v preglednico vpisali pripadajočo stopnjo pokritja (S_p). Stopnja pokritja pomeni, v kolikšni meri so funkcije, procesi, metode itd. poznane, dokumentirane in ali jih obvladujemo, da ne bodo postala vir tveganja. Tako so bili viri tveganj, ki smo jih ocenili z nizko stopnjo pokritja, osnova za podrobnejšo identifikacijo tveganj, povezanih z virom. Npr., če so funkcijske zahteve projekta obdelane v celoti, vpišemo S_p = 100 %.

Slika 18: Radarska slika splošne tveganosti projekta

Če ocenimo, da so obdelane samo polovično, vpišemo S_p = 50 %. Sedaj lahko presodimo, ali je to za nas sprejemljivo, ali moramo funkcijske zahteve revidirati in izpopolniti.

Zmnožek utežne vrednosti D in stopnje pokritja S_p nam da delež pokritja pri tveganju D_p. Na ta način dobimo, na podlagi podatkov (Priloga 1), radarsko sliko splošne tveganosti projekta (Slika 18).

Takšen radarski diagram nazorno pokaže, na katerih področjih imamo največji delež preostalega (nepokritega) tveganja (angl. *Residual risk*). V našem primeru je bil projekt do četrte faze že dodobra utečen in stabiliziran, tako da ni bil več izpostavljen splošnim (najpogostejšim) virom tveganj.

3.2.3 Identifikacija tveganj

Ker smo na podlagi splošne ocene tveganosti projekta ugotovili, da na projektu v četrti fazi pravzaprav ne pričakujemo kakšnih splošnih tveganj, smo se osredotočili na identifikacijo tehničnih tveganj (Tabela 11).

Tabela 11: Seznam tehničnih tveganj (faza 4)

Št.	Tveganje	Vzrok	Posledica
F4.1	Optimizacija zmogljivosti.	V tretji fazi smo se veliko ukvarjali z optimizacijami, vendar rezultat še ni povsem zadovoljiv.	V optimizacijo kode bo potrebno vložiti veliko časa. Če ne bo delovalo, kot je potrebno, bomo imeli velike težave – kupci ne bodo zadovoljni s kakovostjo izdelka.
F4.2	Zaščita podatkov pri prenosu in certificiranje pri uporabnikih.	Enostavni mehanizmi zaščite prenosa podatkov ne bodo zadovoljili kupcev. Zahtevnejše sheme bodo obremenjevale procesor še posebej na počasnejših tablicah.	V optimizacijo, preizkušanje in „piljenje“ kode bo potrebno vložiti veliko časa – podaljšanje projekta.
F4.3	Neznane napake na mobilnih platformah.	Tudi v tretji fazi smo ugotovili, da ima grafična knjižnica kljub nadgradnjam in popravkom še vedno precej napak. V tretji fazi smo dobavitelju knjižnice prijavili 40 napak, od tega jih je bilo odpravljenih samo 34. Vseh napak ne poznamo in najverjetneje se bodo kakšne še pojavile.	Glede „odprtih“ napak ne vemo, ali jih bo dobavitelj pravočasno odpravil, ali bomo morali sami najti rešitev in s predelavami kode napako popraviti ali prikriti – podaljšanje projekta.

3.2.4 Kvalitativna analiza tveganj

Na podlagi seznama možnih tveganj (Tabela 11) smo nekatera identificirana tveganja podrobneje razdelali na posamezne elemente oz. aktivnosti (Tabela 12).

Za vsako tveganje smo določili verjetnost pojava tveganja (*P*), vpliv (*I*) in stopnjo ranljivosti (*V*). Pri tem smo za ocenjevanje uporabili kriterije za kvalitativno ocenjevanje tehničnih tveganj (glej pog. 2.2.3.4).

V ocenjevanje smo vključili tri razvijalce, ki so bili v četrti fazi projekta tudi udeleženi. Ocene, ki smo jih dobili od razvijalcev, so prikazane v prilogi 2. Posamezna tveganja smo na podlagi dobljenih ocen razvrstili od najbolj do najmanj tveganega (Tabela 12).

Tabela 12: Kvalitativna in skupna ocena tveganj

Rang	Tveganje	Ocena tveganja
1	F4.1.1 Prikaz grafičnih komponent bo prepočasen na manj zmogljivih platformah, ali se bodo pojavljale težave pri skaliranju zaradi različnih ločljivosti ciljih naprav.	336
2	F4.1.3 Ponavljanje podatkovnih paketov v primeru izpada, zaradi česar bo potrebno oceniti vpliv na kakovost prikaza.	200
3	F4.3.2 Neznane napake na sistemu iOS, povzročene zaradi uporabe grafične knjižnice.	180
4	F4.3.3 Neznane napake na sistemu Windows, povzročene zaradi uporabe grafične knjižnice.	125
5	F4.1.6 Preveliko število grafičnih elementov, ki bi jih uporabnik aplikacije želel hkrati uporabiti. Ni znano, kako bo to vplivalo na preglednost na različnih ločljivostih in različno zmogljivih platformah.	120
6	F4.2 Potreba po zaščiti podatkov med prenosom. Ni znano, kakšne mehanizme bo potrebno uporabiti in kakšne so zahteve različnih kupcev.	120
7	F4.3.1 Neznane napake na sistemu Android, povzročene zaradi uporabe grafične knjižnice.	80
8	F4.1.2 Procesiranje podatkovnih paketov, predvsem na manj zmogljivih mobilnih napravah.	64
9	F4.1.4 Polnjenje in praznjenje vmesnega pomnilnika, ni znano kakšna velikost vmesnega pomnilnika bo potrebna, še posebej na manj zmogljivih mobilnih napravah ter kakšna strategija bo potrebna v primeru, če bo prihajalo do prekoračitev pomnilnika.	4
10	F4.1.5 Preklop med stranmi, ali bo preklapljanje med posameznimi stranmi imelo vpliv na zvezen pretok podatkov in posodobitev instrumentov. Ali je potrebno navidezno risanje v ozadju, da ob preklopu ne pride do preskokov prikazanih vrednosti.	2

Kot je razvidno iz zgornje tabele, so razvijalci ocenili, da je največje tveganje pričakovati pri prikazu grafičnih komponent in pri optimizaciji kode, ki bo zagotovila, da bo prikaz grafičnih komponent tekoč in dovolj hiter tudi na manj zmogljivih mobilnih napravah.

Nadalje vidimo, da so bila nekatera tveganja ocenjena zelo nizko (npr. F4.1.2, F4.1.4 in F4.1.5) in na tem mestu je smiselno podati tudi neko mejno vrednost za tveganja, s katerimi se bomo aktivno oz. pasivno ukvarjali.

Pri našem projektu smo postavili to mejo na 80. Ocena mejne vrednosti je predvsem pomembna za proces kontroliranja tveganj, saj tako razpoložljive vire osredotočamo na tista

tvegana področja oz. aktivnosti, ki imajo največji vpliv na uspešen potek in zaključek projekta.

3.2.5 Kvantitativna analiza tveganj

Kot smo opisali v poglavju 2.2.4, nas pri kvantitativni analizi tveganj zanimajo predvsem štirje aspekti:

1. vpliv na stroške projekta;
2. vpliv na časovni potek projekta;
3. vpliv na kakovost rezultata projekta;
4. vpliv na izpolnitev projektnih zahtev.

V našem primeru, na projektu X, nismo imeli večjih neposrednih investicij v sredstva, razen nakupa licence za grafično knjižnico in plačila letne članarine, ki omogoča distribucijo aplikacije skozi spletno prodajalno aplikacij. Vpliv tveganj na našem projektu X se je odražal predvsem v časovnem poteku, tj. v zamudi in posledično z višjimi stroški izvedbe.

Zato smo tveganja, ki smo jih identificirali in kvalitativno ocenili s pomočjo razvijalcev v prejšnjem poglavju in do katerih bi lahko prišlo v primeru nastopa tveganja, izrazili z dnevi zamude. Pri tem smo izhajali iz predpostavke, da bodo ob zaključku projekta izpolnjene vse funkcijske zahteve, saj kvantitativno nismo mogli oceniti denarne vrednosti pomanjkljivega funkcijskega obsega (npr. tveganje manjše prodaje zaradi slabše kakovosti ali nepopolne izvedbe).

Kot verjetnost nastanka tveganja smo upoštevali mediano verjetnosti P (Priloga 2), izraženo v odstotkih (npr. ocena 10 → p = 100 %). Morebitne zamude zaradi dodatnih del (Tabela 13), povzročenih z nastankom tveganja, smo podali kot tritočkovni razpon in sicer kot:

- **optimistično** oceno, v nadaljevanju BC (angl. *Best case*);
- **realno** oceno, v nadaljevanju RE, (angl. *Real case*);
- **pesimistično** oceno, v nadaljevanju WC (angl. *Worst case*).

Iz ocen smo izračunali uteženo srednjo vrednost WA (angl. *Weighted average*) s pomočjo naslednje tritočkovne formule:

$$WA = \frac{(BC + 4 \times RE + WC)}{6}$$

(5)

Uteženo srednjo vrednost dni zamude zaradi dodatnih del, kot posledico nastopa tveganja, smo pomnožili s ceno delovnega dneva C (angl. *Cost*). V našem primeru je bila cena delovnega dneva 1.200,00 EUR. Dobljeno ceno zamude smo pomnožili z verjetnostjo nastanka tveganja P (Formula 6). Na ta način smo izračunali pričakovano denarno vrednost posameznega tveganja, v nadaljevanju EMV (angl. *Expected monetary value*).

$$EMV = P \times WA \times C$$

(6)

Pričakovana denarna vrednost da oceno potrebne denarne rezerve zaradi identificiranih tveganj in verjetnosti njihovega pojava, skupno 64.800,00 EUR (Tabela 13).

Če bi se posamezno tveganje uresničilo in bi bilo potrebno odpravljati posledice, bi bila cena odprave posledice v najboljšem primeru enaka BC x C, realno RE x C in v najslabšem primeru WC x C. Torej, če bi se na našem projektu materializirala vsa identificirana tveganja v najslabše ocenjeni obliki, bi to pomenilo 147.600,00 EUR dodatnih stroškov, v nadaljevanju WCC (angl. *Worst case costs*) (Tabela 13, Formula 7).

$$WCC = \sum_{i=1}^{n} WC_i \times C$$

(7)

Tabela 13: Kvantitativna ocena tveganja

#	**Tveganje**	**P**	**BC**	**RE**	**WC**	**WA**	**EMV (€)**	**WCC (€)**
F4.1.1	Prikaz grafičnih komponent	0,8	10	12	15	12,2	11.680,00	18.000,00
F4.1.3	Ponavljanje pod. paketov	0,8	12	16	20	16,0	15.360,00	24.000,00
F4.3.2	Neznane napake na sistemu iOS	0,6	10	12	14	12,0	8.640,00	16.800,00
F4.3.3	Neznane napake na sist. Windows	0,5	10	12	14	12,0	7.200,00	16.800,00
F4.1.6	Preveliko število gr. elementov	0,4	5	8	10	7,8	3.760,00	12.000,00
F4.2	Zaščita podatkov med prenosom	0,4	20	30	40	30,0	14.400,00	48.000,00
F4.3.1	Neznane napake na sistemu Android	0,4	5	8	10	7,8	3.760,00	12.000,00
	Skupaj						64.800,00	147.600,00

Po oceni pričakovane denarne vrednosti (EMV) se je spremenila tudi prioriteta tveganj (Tabela 14). Zaščita podatkov med prenosom je poskočila iz šestega na drugo mesto, medtem ko sta prvotno najvišje uvrščeni tveganji v procesu kvalitativne analize prvotno visoko uvrstitev obdržali.

Iz obeh tabel (Tabela 13 in Tabela 14) je razvidno, da so stroški odprave posledic visoki pri tveganju, povezanim z zaščito podatkov med prenosom (F4.2).

Prav tako so stroški odprave posledic visoki tudi pri tveganjih neznanih napak na sistemih iOS in Windows (F4.3.2, F4.3.3).

Tabela 14: Popravljena prioriteta tveganj glede na pričakovano denarno vrednost (EMV)

Prio.	Tveganje	Ocena tveganja	Verjetnost p	EMV (€)	Nova prioriteta
1	F4.1.1 Prikaz grafičnih komponent	336	0,8	11.680,00	3 ↓
2	F4.1.3 Ponavljanje podatkovnih paketov	200	0,3	15.360,00	1 ↑
3	F4.3.2 Neznane napake na sistemu iOS	180	0,6	8.640,00	4
4	F4.3.3 Neznane napake na sistemu Windows	125	0,5	7.200,00	5
5	F4.1.6 Preveliko število grafičnih elementov	120	0,4	3.760,00	6
6	F4.2 Zaščita podatkov med prenosom	120	0,4	14.400,00	2 ↑↑
7	F4.3.1 Neznane napake na sistemu Android	80	0,4	3.760,00	7

Na osnovi primerjave lahko sklepamo, da bi postopek kvantitativne analize lahko poenostavili tako, da bi pričakovano denarno vrednost tveganja (EMV) računali iz najslabše ocene odprave posledic tveganja, v nadaljevanju EWCC (angl. *Expected Worst Case Costs*) in ne iz srednjih vrednosti, po naslednji formuli:

$$EWCC = P \times WCC \tag{8}$$

Na podlagi tega dobimo tudi bolj konzervativno oceno potrebnih denarnih rezerv za pokrivanje najslabših možnih scenarijev, ki znaša v tem primeru 80.880,00 EUR.

Tabela 15: Popravljena prioriteta tveganj glede na EWCC

Prio.	Tveganje	Verjetnost p (%)	WCC (€)	EWCC (€)
1	F4.2 Zaščita podatkov med prenosom	0,4	48.000,00	19.200,00
2	F4.1.3 Ponavljanje podatkovnih paketov	0,8	24.000,00	19.200,00
3	F4.1.1 Prikaz grafičnih komponent	0,8	18.000,00	14.400,00
4	F4.3.2 Neznane napake na sistemu iOS	0,6	16.800,00	10.080,00
5	F4.3.3 Neznane napake na sistemu Windows	0,5	16.800,00	8.400,00
6	F4.3.1 Neznane napake na sistemu Android	0,4	12.000,00	4.800,00
7	F4.1.6 Preveliko število grafičnih elementov	0,4	12.000,00	4.800,00
	Skupaj		**147.600,00**	**80.880,00**

Iz rezultatov kvantitativne analize lahko sklepamo, da:

- predstavlja največje tveganje zaščita podatkov med prenosom, ponavljanje podatkovnih paketov v primeru izpada in prikaz grafičnih komponent;
- potrebujemo varnostno rezervo v višini 64.800,00 EUR, če upoštevamo srednjo vrednost ocen odprave posledic materializacije tveganj oz. 80.880,00 EUR, če upoštevamo najslabše ocene;
- to pomeni, preračunano na dneve zamude realno 54 dni oz. v najslabšem primeru 67,4 dni zamude pri dokončanju projekta (zaradi odprave posledic materializacije vseh tveganj, ki smo jih vključili v terminski plan projekta).

3.2.6 Planiranje rezerv

Kot smo povedali v teoretičnem delu (pog. 2.2.6), potrebujemo dve vrsti rezerv:

- **projektne;**
- in **varnostne rezerve**.

Projektne rezerve v našem primeru predstavljajo rezerve v negotovosti ocen potrebnih del (angl. *Management reserve*). Kot smo opisali v pog. 1.2, nam negotovost pri planiranju v oceno obsega del vnaša raven organizacijske zrelosti. V našem primeru lahko trdimo, da je naša organizacija na 3. nivoju CMMI modela, saj ima formalizirane procese in certifikat kakovosti ISO 9001, nima pa vpeljane metrike za merjenje učinkovitosti procesov.

Tako smo zaradi negotovosti pri oceni potrebnih del upoštevali 30 % rezervo, kar je tudi v skladu z ugotovitvami De Marca in Lister-ja (2003, str. 91–94), Eberta (2012, str. 152–157 in 294–297) in Dua (2012, str. 5–6). Kot varnostno rezervo (angl. *Contingency reserve*) smo pri projektu upoštevali izračunano vrednost EWCC (80.880,00 EUR), ki smo jo izračunali po formuli (8) v prejšnjem poglavju.

Pri tem je potrebno poudariti, da je pri predstavitvi stroškovnika naročnikom projekta smiselno prikazati izračun v tabelarični obliki (Tabela 13), v kateri so prikazane pričakovane denarne vrednosti (EMV) in stroški odprave posledic materializacije tveganja v najslabšem možnem primeru (WCC oz. EWCC).

Glede na to, da je bila vrednost četrte faze z vključeno 30 % projektno rezervo ocenjena na 156.000,00 EUR, predstavlja ocenjena varnostna rezerva 52 % vrednosti projekta. Iz tega

lahko sklepamo, da gre za visoko tvegani projekt, kar potrjuje tudi ocenjena prekoračitev stroškov projekta za 195 % v primeru, da bi prišlo do materializacije vseh predvidenih tveganj v najslabši možni obliki.

Tabela 16: Stroškovnik projekta z vključeno projektno in varnostno rezervo

	Znesek	**Faktor**
Strošek projekta po realni oceni potrebnega obsega del	120.000,00 €	
Projektna rezerva zaradi negotovosti (»buffer«)	36.000,00 €	0,30
Skupaj (PC)	**156.000,00 €**	
Varnostna rezerva (EWCC)	80.880,00 €	0,52
Potreben proračun (PC + EWCC)	236.880,00 €	
Potreben proračun v najslabšem možnem primeru (PC + WCC)	303.600,00 €	1,95

Začetna (projektna) varnostna rezerva je v mejah priporočil, tj. 30–70 % (Dua, 2012, str. 5–6), kar potrjuje sprejemljivost naše ocene. Pri tem je potrebno poudariti, da gre za oceno začetne varnostne rezerve, ki se lahko tekom projekta, v procesu kontroliranja tveganj spreminja na podlagi sprotnih ugotovitev. V primeru učinkovitega managementa tveganj pričakujemo, da se bo potrebna varnostna rezerva tekom projekta zmanjševala.

S pomočjo stroškovnika (Tabela 16) naročnike projekta seznanimo s:

- skupnim stroškom projekta z vključeno projektno rezervo, ki jo upoštevamo zaradi negotovosti pri oceni potrebnega obsega del;
- potrebnim proračunom z vključeno varnostno rezervo, za odpravo posledic tehničnih tveganj ob materializaciji le-teh;
- potrebnim proračunom v primeru materializacije vseh tveganj v najslabši možni obliki.

Na podlagi tako pripravljenega stroškovnika se naročniki projekta lažje odločijo, ali so projekt pripravljeni financirati, saj imajo celovitejšo sliko in so vnaprej seznanjeni z možnimi posledicami zaradi nastopa tveganj.

3.2.7 Ukrepi za znižanje tveganosti

Kot smo povedali v pog. 2.2.5, se na negativna tveganja lahko odzovemo z: izogibanjem, zmanjšanjem, prenosom ali sprejetjem tveganja. Na priložnosti se lahko odzovemo z: izrabo, delitvijo, povečanjem in sprejetjem priložnosti.

Zato smo se tekom projekta aktivno ukvarjali samo s tveganji, katerih trenutno ocenjena vrednost (R) je presegala vrednost 80 točk (Tabela 17). Za ta tveganja smo uporabili strategijo zmanjšanja učinka in verjetnosti pojava. Ostala tveganja smo spremljali samo pasivno (sprejetje tveganja).

Za tveganje F4.1.6. (preveliko število grafičnih komponent) smo omejili največje število grafičnih elementov, ki jih lahko uporabnik v okviru konfiguracije doda na posamezno stran. To smo potem tudi zapisali v navodila za uporabo, kot znano funkcijsko omejitev pri postavitvi konfiguracije.

Za ostali dve tveganji: F4.3.2 in F4.3.3, ki se nanašata na neznane napake na operacijskih sistemih Windows in iOS, smo poskušali zmanjšati verjetnost pojava, saj učinka nismo mogli oceniti neposredno. To smo dosegli tako, da smo investirali več časa, kot je bilo načrtovano, v izdelavo avtomatiziranih preizkusov (rast napovedi ob zaključku, EAC v kontrolnih točkah M1–M2 in M3–M4, Slika 20).

Enako metodo smo uporabili tudi za najvišje ocenjeno tveganje F4.1.1., ker smo ocenili, da se temu tveganju na počasnejših mobilnih napravah ne bomo mogli izogniti vse do konca.

3.2.8 Kontroliranje tveganj

Kot smo omenili v pog. 3.2.2, je bil v četrti fazi projekt stabiliziran že do te mere, da se s splošnimi projektnimi tveganji nismo več ukvarjali. Zato smo enkrat mesečno, v kontrolnih točkah M_x, skupaj z razvijalci revidirali seznam in prioriteto tehničnih tveganj (Tabela 11 in Tabela 12).

Popravljene ocene tehničnih tveganj smo v petih kontrolnih točkah (mesecih) (Tabela 17 in Priloga 3) spremljali v grafični obliki. Pri tem smo se poslužili grafičnega prikaza (Slika 19) zmanjšanja ocen tveganj skozi čas (angl. *Risk burndown chart*). Glede na popravljene ocene verjetnosti tveganj (Priloga 3) smo v vsaki kontrolni točki popravili (zmanjšali) tudi potrebno varnostno rezervo (Tabela 18 in Tabela 19).

Tabela 17: Popravljene ocene tveganj tekom projekta

#	Tveganje	M0	M1	M2	M3	M4	M5
F4.1.1	Prikaz grafičnih komponent	336	252	150	126	60	0
F4.1.3	Ponavljanje paketov v primeru izpada	200	200	200	200	50	0
F4.3.2	Neznane napake na sistemu iOS	180	180	180	100	40	0
F4.3.3	Neznane napake na sistemu Windows	125	100	80	60	24	0
F4.1.6	Preveliko število grafičnih elementov	120	60	0	0	0	0
F4.2	Potreba po zaščiti podatkov	120	120	0	0	0	0
F4.3.1	Neznane napake na sistemu Android	80	80	60	20	20	0

Legenda: M0 .. M5 – Kontrolne točke (angl. *Milestones*)

Tabela 18: Spreminjanje varnostne rezerve med M1 in M2

		M1		M2	
Tveganje	**WCC (€)**	**p**	**EWCC (€)**	**p**	**EWCC (€)**
F4.2 Zaščita podatkov	48.000,00	0,4	19.200,00	0,0	0,00
F4.1.3 Ponavljanje paketov	24.000,00	0,8	19.200,00	0,8	19.200,00
F4.1.1 Prikaz grafičnih komponent	18.000,00	0,6	10.800,00	0,4	7.200,00
F4.3.2 Neznane napake iOS	16.800,00	0,4	6.720,00	0,4	6.720,00
F4.3.3 Neznane napake Windows	16.800,00	0,5	8.400,00	0,5	8.400,00
F4.3.1 Neznane napake Android	12.000,00	0,4	4.800,00	0,3	3.600,00
F4.1.6 Preveliko število graf. elem.	12.000,00	0,3	3.600,00	0,0	0,00
Skupaj			**72.720,00**		**45.120,00**

Slika 19: Spreminjanje kvalitativnih ocen tveganj skozi potek projekta

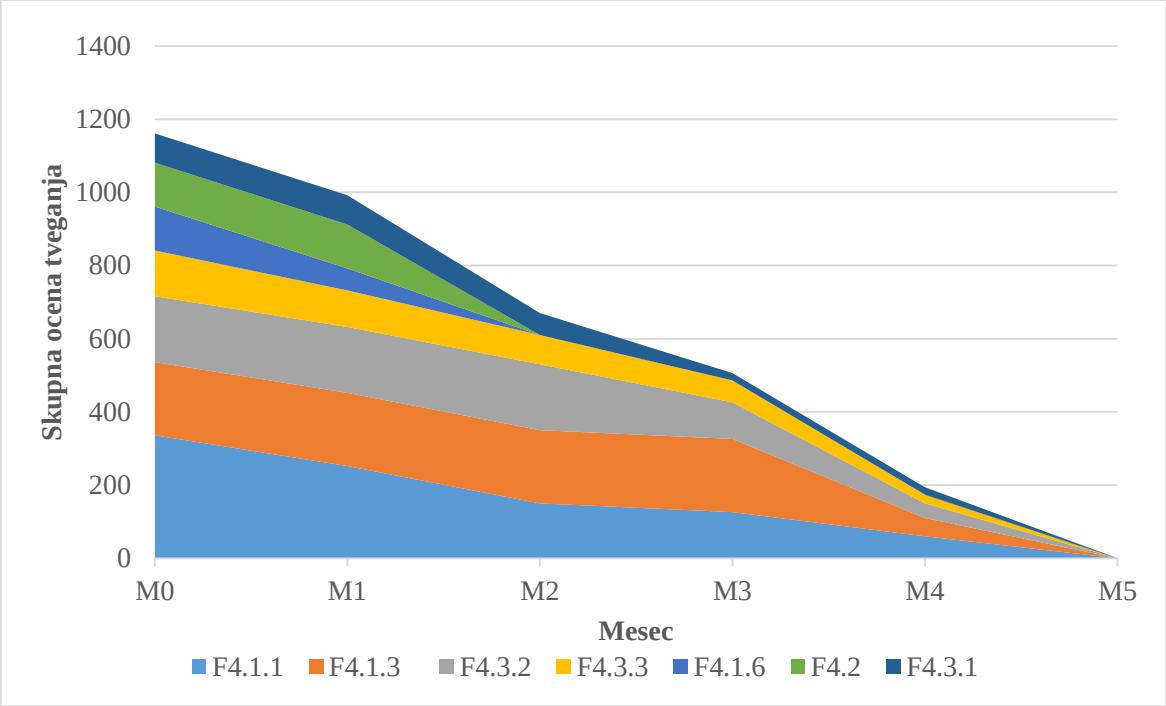


Tabela 19: Spreminjanje varnostne rezerve med M3 in M4

		M3		M4	
Tveganje	**WCC (€)**	**p**	**EWCC (€)**	**p**	**EWCC (€)**
F4.2 Zaščita podatkov	48.000,00	0,0	0,00	0,0	0,00
F4.1.3 Ponavljanje paketov	24.000,00	0,8	19.200,00	0,2	4.800,00
F4.1.1 Prikaz grafičnih komponent	18.000,00	0,3	5.400,00	0,2	3.600,00
F4.3.2 Neznane napake iOS	16.800,00	0,3	5.040,00	0,2	3.360,00
F4.3.3 Neznane napake Windows	16.800,00	0,5	8.400,00	0,5	8.400,00
F4.3.1 Neznane napake Android	12.000,00	0,2	2.400,00	0,2	240,00

F4.1.6 Preveliko število gr. elem.	12.000,00	0,0	0,00	0,0	0,00
Skupaj			**40.440,00**		**20.400,00**

3.2.9 Analiza prislužene vrednosti

Analiza prislužene vrednosti (angl. *Earned value analysis*) je zelo učinkovito orodje za sprotno spremljanje projekta in za zgodnje odkrivanje morebitnih skritih tveganj (Nagrecha, 2002).

Tudi mi smo poskušali ugotoviti, ali se na projektu pojavljajo kakšni zaostanki, ki so povzročeni s skritimi tveganji, ki jih še nismo identificirali in, ali so morebitni zaostanki na projektu posledica negotovosti pri načrtovanja projekta. V ta namen smo tedensko spremljali stroške projekta ACWP[5] (angl. *Actual cost of work performed*) in prisluženo vrednost BCPW[6] (angl. *Budgeted cost of work performed*).

Prisluženo vrednost (BCPW) smo ocenjevali na podlagi tedenskih poročil razvijalcev o opravljenih urah in napredku pri posameznih delovnih nalogah (angl. *Work packages*). Iz tega smo dobili indeks stroškovne učinkovitosti (Formula 9), v nadaljevanju CPI (angl. *Cost performance index*). Stroškovna učinkovitost nam pove, koliko vrednosti dobimo za vložena denarna sredstva. Če je indeks CPI večji od 1, to pomeni, da projekt stane manj kot smo planirali in obratno, da bo stal več, če je CPI manjši od 1.

$$CPI = \frac{BCWP}{ACWP} \tag{9}$$

S sprotnim tedenskim računanjem napovedi ob zaključku projekta, v nadaljevanju EAC *(*angl. *Estimate at completion*), lahko zelo hitro ocenimo, kaj se dogaja s projektom.

Napoved ob zaključku projekta smo v kontrolnih točkah računali po naslednji formuli:

$$EAC = \frac{BAC}{CPI} \tag{10}$$

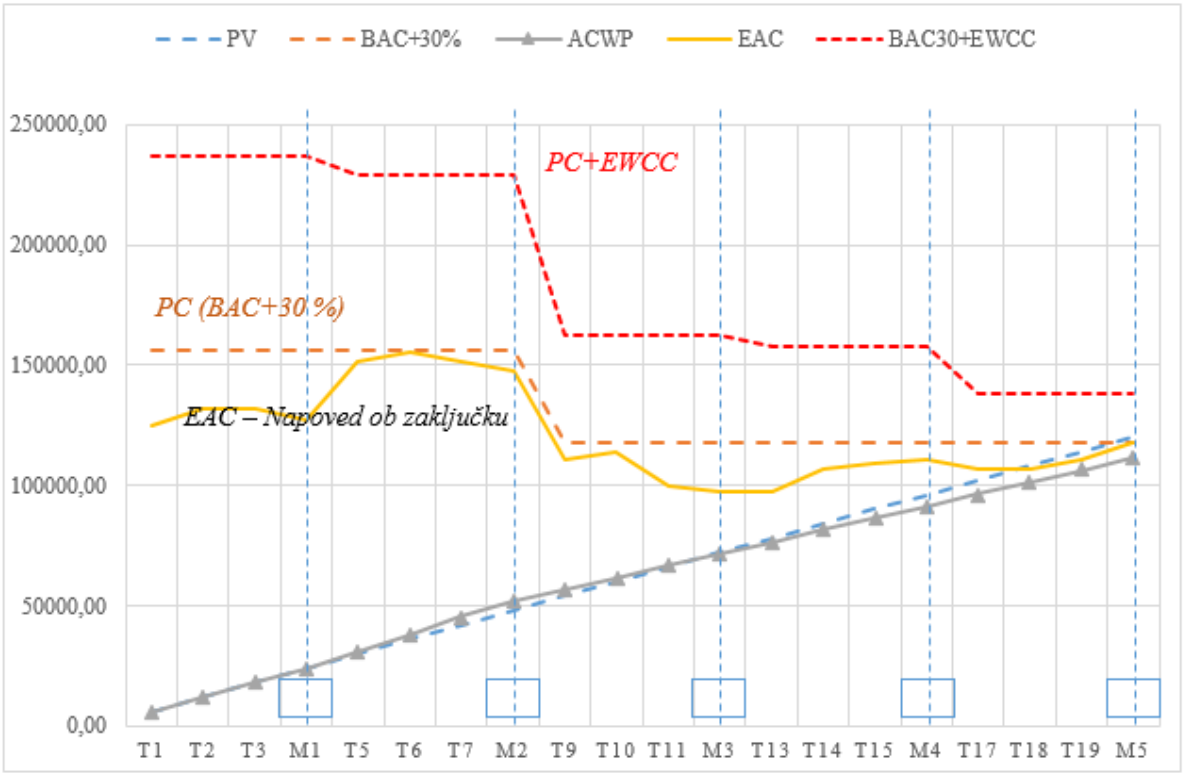
[5] Alternativno AC (angl. *Actual cost*)
[6] Alternativno EV (angl. *Earned value*)

Vrednost BAC (angl. *Budget at completion*) je planirana vrednost projekta brez projektnih rezerv[7] in služi kot orientacijska, izhodiščna vrednost za kontroliranje projekta (Formula 10). V našem primeru smo vrednost projekta (BAC) razdelili na 20 tednov in tako dobili referenčno planirano vrednost PV (angl. *Planned value*) za posamezni teden (Slika 20, Priloga 4).

Da bi imeli boljšo predstavo, v kakšnih mejah se odvija projekt, smo na graf dodali še planirani strošek projekta, povečan za projektno rezervo (v nadaljevanju PC) ter strošek projekta s projektno in varnostno rezervo za pokritje tveganj (PC + EWCC, Slika 20).

Kot vidimo iz poteka EAC (Slika 20, Priloga 4), je vrednost med kontrolno točko M1 in M2 znatno poskočila. Razlog za to leži v tem, da smo angažirali dodatnega razvijalca, ki je pripravil testno okolje za preizkušanje grafičnih komponent. Vrednost projekta smo zmanjšali v kontrolni točki M2, saj smo po razgovorih s ključnimi strankami prišli do ugotovitve, da ne potrebujejo zaščite podatkov med prenosom (F4.2, pog. 3.2.3) in da lahko omejimo število grafičnih komponent na vrednost, ki ne bo predstavljala težav (F4.1.6, pog. 3.2.4).

Slika 20: Tedensko spremljanje in napovedi stroškov ob zaključku projekta



3.2.10 Učinkovitost ukrepov za znižanje tveganosti

[7] Uporaba BAC brez projektne rezerve je kot orientacijsko izhodišče primernejša, ker v tem primeru vidimo, ali porabljamo tudi projektno rezervo.

Verjetnost pojava omenjenih tveganj smo poskušali zmanjšati tako, da smo ključne grafične elemente in algoritme razvijali posebej kot prototipne module in optimizirali kodo za vsak element posebej. Modularna izvedba in sprotno preizkušanje komponent je imelo ugoden vpliv na končno preizkušanje, ki smo ga izvedli v planiranem roku. Vsi ti preventivni ukrepi so zahtevali dodatne vire (strošek dodatnega razvijalca), ki smo jih med točkama M1 in M2 črpali iz projektne rezerve. Na projekt je ugodno vplivala tudi izključitev dveh najvišje ocenjenih tveganj po točki M2.

Če teh tveganj ne bi identificirali in se z njimi aktivno ukvarjali (v smislu revidiranja projektnih zahtev), bi zagotovo prekoračili prvotno načrtovane stroške projekta.

Četrta faza je bila tako izvedena v predvidenem roku, vendar zgolj na račun dejstva, da smo lahko zmanjšali prvotno planirani obseg del iz ocenjenih 120.000,00 EUR na 109.600,00 EUR in izločili dve najvišje ocenjeni tveganji, ki bi v primeru materializacije (Tabela 15) povzročili najmanj 72.000,00 EUR dodatnih stroškov.

Končni strošek projekta je ob zaključku znašal tako 105.600,00 EUR oz. dobrih 4 % manj, kot je bilo planirano po korekciji v kontrolni točki M2. Pri tem je potrebno ponovno poudariti, da je bil v četrti fazi projekt že dodobra stabiliziran ter da je bilo vsaj v začetku odstopanje napovedi stroškov projekta EAC zelo na meji.

3.3 Povzetek analize projekta

Kot je razvidno iz analize managementa tveganj na projektu X, smo se kljub sistematičnemu pristopu k identifikaciji, kvalitativni in kvantitativni oceni tveganj srečali z naslednjimi težavami:

1. Kljub temu, da je bil projekt v četrti fazi že dodobra stabiliziran in so bili razvijalci s tehničnimi in projektnimi zahtevami dokaj dobro seznanjeni, je bila prvotna ocena tveganosti dokaj visoka in s tem tudi obseg potrebne varnostne rezerve (52 %). Za takšen relativno kratek in pregleden projekt je to še vedno veliko.

   V našem primeru je bilo zato zelo težko prepričati naročnike projekta, da je takšna rezerva resnično potrebna, saj so bili ti mnenja, da razvijalci s svojimi ocenami tveganosti in še posebej pri oceni potrebnih del pretiravajo. Tukaj se kaže navzkrižje interesov in izpostavi naloga managerja projekta. Ta mora v prvi vrsti izločiti pristranskost pri ocenah tveganja (glej pog. 2.2.3.1 in 2.2.3.2) in na podlagi kvalitativno in kvantitativno podprtih ocen naročnikom projekta argumentirati plan projekta. V takšnih primerih namreč tvegamo, da projekt ne bo sprejet ali pa naročniki projekta prisilijo managerja projekta k optimizaciji stroškov.
   Praksa kaže, da optimizacija ali strah pred prekoračitvijo stroškov negativno vplivata na funkcijski obseg in kakovost (glej pog. 2.1), ki lahko sproži sekundarna tveganja

(nezadovoljstvo kupcev izdelka oz. naročnikov projekta, visoke stroške odprave napak itd.).

2. Pri metodi analize prislužene vrednosti se po eni strani kaže uporabnost metode za kvantitativno podprto spremljanje projekta z dobro indikacijo težav. Manager projekta jo lahko uporabi za pravočasno komuniciranje težav na projektu in sproži korekturne akcije. Pravočasno komuniciranje težav na projektu je dosti ugodnejše, kot pa prikrivanje in izvajanje (napačnih) ali ne-izvajanje korekturnih akcij.

   Po drugi strani pri razvojnih projektih programske opreme ni enostavno oceniti prisluženo vrednost v določeni kontrolni točki, saj je le-to težko izmeriti. V določenem trenutku razvijalec namreč težko pove, kolikšen odstotek delovne naloge je izveden, saj kot smo v nalogi že večkrat omenili, ni mogoče vnaprej predvideti potrebnega števila vrstic programske kode za izvedbo določene naloge.

   Zaradi tega je v ocenah napredka prisotna velika negotovost in zaradi le-te lahko v določenem trenutku verjamemo, da je potek projekta boljši, kot je v resnici. Praksa tudi kaže, da razvijalci poročajo o napredku v začetnih fazah projekta bolj optimistično kot proti koncu, ko začne pritiskati končni rok (angl. *Deadline*). Takrat postanejo napovedi v večini primerov bolj pesimistične in po navadi se s strani razvoja začne stopnjevati pritisk na managerja projekta z zahtevami po zmanjšanju funkcijskega obsega, ali zmanjšanju kakovosti.

   Gre predvsem za zahteve po zmanjšanju pokritja preizkusov (angl. *Test coverage*) ali v povišanju sprejemljive meje števila odkritih napak v poznejših fazah preizkušanja (angl. *Software maturity index*). Manager projekta lahko le s proaktivnim sodelovanjem in dnevnim poročanjem dobi realnejšo sliko o stanju na projektu. Šele na višjih stopnjah CMMI modela, z vpeljano metriko, lahko pričakujemo, da so ocene bolj točne in kažejo s tem realnejšo sliko.

SKLEP

Kljub napredku teorije in prakse managementa projektov se pri projektih razvoja vgrajene programske srečujemo z veliko kompleksnostjo izvedbenih nalog in s tem povezanimi težavami.

Management tovrstnih projektov zato zahteva zelo poglobljena tehnična in managerska znanja in predvsem sistematični pristop, ki smo ga poskušali opisati v tej nalogi in podpreti s podrobnejšim pregledom teorije, z namenom, da bi v njej našli razlago za vzroke težav in zbrali napotke za njihovo uspešno reševanje.

Največja težava, s katero se pri tovrstnih projektih srečujemo, je nedvomno, da je v fazi načrtovanja projekta praktično nemogoče načrtovati potreben obseg del. Vnaprej namreč ni mogoče oceniti ali predpisati potrebno število vrstic programske kode za izvedbo določene

projektne naloge, kakor tudi ne medsebojnih vplivov sistemske-programske in strojne opreme.

Pri projektih vgrajene programske opreme smo tako ves čas trajanja projekta v negotovosti, saj dokler niso izdelani in preizkušeni vsi programski moduli in strojna oprema, ne vemo, ali bomo lahko projekt uspešno zaključili. V tem se tudi kaže bistvena razlika med klasičnimi IT projekti, kjer razvijamo programsko opremo na delujočem sistemu.

V tej nalogi ni bilo mogoče potegniti ostre ločnice med obema vrstama projektov, saj za obe vrsti veljajo iste zakonitosti. V primeru vgrajene programske opreme imamo pač opravka z dodatnimi stopnjami kompleksnosti in je zaradi tega potrebno vložiti v pripravo projekta, identifikacijo in oceno tveganj, kakor tudi učinkovito kontroliranje tveganj, toliko več časa in energije. Zato smo poskušali v tej nalogi problematiko managementa tveganj osvetliti od zgoraj navzdol in izpostaviti tiste dejavnike, ki lahko v tovrstne projekte vnašajo tveganja. Ugotovitve iz teorije in metode iz analize projekta X lahko tako apliciramo na obe vrsti projektov.

Na podlagi navedb različnih avtorjev v literaturi in na spletu o stopnji prekoračitve projektnih stroškov in časovnih zamud ter najpogostejših razlogov, ugotavljamo, da je kar 67 % zrelih podjetij, ki razvijajo programsko opremo in so na tretji stopnji poslovne zrelosti po CMMI modelu že prekoračilo stroške izvedbe ali časovne roke v povprečju za 200 %. Iz naših ugotovitev pri projektu X lahko sklepamo, da podjetja najverjetneje ne upoštevajo dovolj negotovosti, ki izhajajo iz ravni poslovne zrelosti in potrebnih varnostnih rezerv ne planirajo dovolj dobro.

Dopuščamo tudi možnost, da se managerji projektov v strahu, da do izvedbe projekta zaradi previsokih izvedbenih stroškov ne bi prišlo, poslužujejo prikrojenih, zmanjšanih ocen. To poskušajo kasneje, tekom projekta, nadoknaditi s pritiskom na sodelavce, še posebej pa na zunanje izvajalce. Slednje problema ne bo rešilo, zagotovo pa negativno vpliva na kakovost izvedbe.

Naslednja težava, ki se kaže v našem podjetju in ki je na 3. stopnji ravni poslovne zrelosti, je neustreznost ali togost predpisanih procesov. Managerji in notranji naročniki projektov tako vztrajajo pri določenih ustaljenih praksah in predpisanih procesih in ne upoštevajo specifičnosti posameznih projektov, kar je po našem mnenju nujno potrebno.

Za vsak projekt posebej je nujno potrebno pretehtati, kakšen razvojni model je najbolj primeren in kakšna tveganja lahko pričakujemo zaradi izbire razvojnega modela. V poglavju 1.5.4 opisan Boehm-ov spiralni model, ki predstavlja nekakšen hibrid linearnih in cikličnih metod, lahko pomaga odpraviti težave pri izbiri razvojnega modela.

Bistvenega pomena za učinkovit management tveganj se nam zdi zavestna odločitev za ali proti managementu tveganj pred začetkom projekta, saj zahteva proces dodatni čas in energijo. Prav tako so lahko podatki, ki jih pridobimo v procesu identifikacije tveganj, kvalitativne in še posebej kvantitativne analize, napačno razumljeni, kar zahteva konsenz med naročniki, managementom in izvajalci projekta. Zato je odprava pristranskosti pri identifikaciji tveganj toliko bolj pomembna.

V analitičnem delu predlagana hitra ocena projektnega tveganja, ki prikaže »radarsko sliko« izpostavljenosti določenih področij projekta splošnim virom tveganj, lahko poda dokaj dobro oceno tveganja in s tem podpre odločitev za ali proti managementu tveganj.

Podrobna in predvsem sprotna identifikacija ter ocena tveganj je osnova za učinkovito kontroliranje projektov in za pravočasno odzivanje in komunikacijo pri težavah na projektu. S kvantitativno analizo podprto načrtovanje varnostih rezerv nam pomaga postaviti dobro finančno konstrukcijo projekta in hkrati osvetliti tveganja z največjim vplivom na projekt.

Kot je razvidno iz analize projekta X, je kljub dobri hitri oceni tveganosti in že stabiliziranem projektu v četrti fazi, smiselno vložiti čas in energijo v management predvsem tehničnih tveganj. Če se z managementom tveganj na projektu ne bi proaktivno ukvarjali, bi projekt zagotovo presegel načrtovane stroške in zamudil rok izvedbe.

Čeprav je bil projekt, kot je razvidno iz analize, zaključen v roku in znotraj predvidenih stroškov, vidimo, da smo se ves čas projekta gibali na meji sprejemljivih stroškov in da smo za izogib tveganjem porabili praktično vso projektno rezervo. S tem smo nazorno pokazali, kako lahko projekt stabiliziramo s sistematičnim pristopom in s pravočasnim reagiranjem na projektna tveganja le-ta preprečimo. Proaktivno ukvarjanje s tveganji, kot smo prikazali na primeru projekta X, pomaga usmerjati managementu projektov energijo tja, kjer je potrebno, s čimer lahko povečamo stroškovno učinkovitost in se izognemo nepotrebnim zapletom.

LITERATURA IN VIRI

1. Addison, T., & Vallabh S. (2002). Controlling Software Project Risks – an Empirical Study of Methods used by Experienced Project Managers. Najdeno 20. aprila 2014 na spletnem naslovu http://www.itu.dk/people/katten/speciale/Controlling_Software_Project_Risks_an_Empirical_Study_of_Methods.pdf

2. Ambler, S. (2014). Examining the Agile Cost of Change Curve. Najdeno 28. junija 2014 na spletnem naslovu http://www.agilemodeling.com/essays/costOfChange.htm

3. Arnuphaptrairong, T. (2011). Top Ten Lists of Software Project Risks: Evidence from the Literature Survey. Najdeno 20. maja 2014 na spletnem naslovu http://www.uio.no/studier/emner/matnat/ifi/INF5181/h14/pensumliste/microsoft-word---iaeng-top-ten-lists-of-software-project-risk1---imecs2011_pp732-737.pdf

4. Bach, J. (1999). The Immaturity of CMM. Najdeno 28. junija 2014 na spletnem naslovu http://www.satisfice.com/articles/cmm.shtml

5. Berg, A. (2015). The Magic Triangle and Devil's Quadrangle – Understanding Project Management Models. Najdeno 9. septembra 2015 na spletnem naslovu http://www.inloox.com/company/blog/articles/the-magic-triangle-and-devil-s-quadrangle-understanding-project-management-models

6. BITKOM. (2010). Eingebettete Systeme – Ein strategisches Wachstumsfeld für Deutschland, Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V. Najdeno 20. junija 2014 na spletnem naslovu http://www.bitkom.org/files/documents/EingebetteteSysteme_web.pdf

7. Blank, S. (2014). Tesla and Adobe: Why Continous Deployment May Dissapoint Customers. Najdeno 1. junija 2014 na spletnem naslovu http://www.inc.com/steve-blank/tesla-and-adobe-why-continuous-deployment-may-mean-continuous-customer-disappointment.html

8. Boehm, B. (1989). *Software Risk Management*. Washington: Computer Society Press.

9. Boehm, B. (2000). Project Termination Doesn't Equal Project Failure. Najdeno 15. januarja 2014 na spletnem naslovu http://www.cs.unc.edu/~welch/class/comp145/media/docs/Boehm_Term_NE_Fail.pdf

10. DeMarco, T., & Lister, T. (2003). *Waltzing with Bears: Managing Risk on Software Projects*. New York: Dorset House Publications

11. Dua, R. (2012). Contingency Versus Management Reserve. Micro Planning International Australia. Najdeno 17. marca 2014 na spletnem naslovu http://www.microplanning.com.au/wp-content/uploads/downloads/2012/01/Contingency.pdf

12. Ebert, C. (2012). *Systematisches Requirements-Engineering und Management: Anforderungen ermitteln, spezifizieren, analysieren und verwalten*. Heidelberg: Dpunkt-Verlag.

13. Ebert, C., & Jones, C. (2009). Embedded Software: Facts, Figures and Future. Najdeno 17. aprila 2014 na spletnem naslovu http://www6.in.tum.de/pub/Main/TeachingWs2013MSE/embeddedSoftwareTrend.pdf

14. Fowler, M., & Highsmith, J. (2001). The Agile Manifesto. Najdeno 5. aprila 2014 na spletnem naslovu http://www.pmp-projects.org/Agile-Manifesto.pdf

15. Graham, D. (1990). Incremental Development and Delivery for Large Software Systems. Najdeno 3. aprila 2014 na spletnem naslovu http://hem.bredband.net/andule/evo-gilb/graham1.htm

16. Grady, R. (1992). *Practical Software Metrics for Project Management and Process Improvement*. Englewood Cliffs: Prentice Hall.

17. Grand View Research. (2014). Global Embedded Systems Market By Product (Hardware, Software), By Application (Automotive, Consumer Electronics, Healthcare, Industrial, Military and Aerospace, Telecommunication) Expected to Reach USD 214.39 Billion by 2020. Najdeno 1. septembra 2014 na spletnem naslovu http://www.grandviewresearch.com/press-release/global-embedded-systems-market

18. Hopkinson, M. (2011). *The Project Risk Maturity Model: Measuring and Improving Risk Management Capability*. Farnham, Surrey: Gower Publishing Ltd.

19. Humprey, S. (1989). *Managing the Software Process*. Reading, MA: Addision-Wesley.

20. IABG. (2006). Grundlagen des V-Modells. Najdeno 18. marca 2014 na spletnem naslovu http://v-modell.iabg.de/index.php?option=com_docman&task=doc_view&gid=48

21. Janes, A., & Succi, G. (2014). *Lean Software Development in Action*. London: Springer Verlag.

22. Jones, C. (2012). A Short History of the Lines of Code (LOC) Metric. Najdeno 3. januarja 2015 na spletnem naslovu http://http://www.ifpug.org/Documents/Jones-LinesofCodeMetricV6.pdf

23. Kerzner, H. (2001). *Strategic Planning for Project Management using a Project Management Maturity Model*. New York: Wiley & Sons.

24. Koopman, P. (2010). Risk Areas in Embedded Software Industry Projects. Najdeno 18. aprila 2014 na spletnem naslovu http://users.ece.cmu.edu/~koopman/pubs/koopman10_risk_areas_embedded_projects.pdf

25. Koopman, P. (2011). Avoiding the Top 43 Embedded Software Risks. Najdeno 21. februarja 2014 na spletnem naslovu http://www.ece.cmu.edu/~koopman/pubs/koopman11_escsv_handouts.pdf

26. Kuster, J., Huber, E., Lippmann, R., Schmid, A., Schneider, E., Witschi, & Wüst, R. (2011). *Handbuch Projektmanagement*. Berlin: Springer Verlag.

27. McConnel, S. (1996). Classic Mistakes Enumerated. Najdeno 25. aprila 2014 na spletnem naslovu http://www.stevemcconnell.com/rdenum.htm

28. McManus, J. (2004). *Risk Management in Software Development Projects*. Burlington, MA: Elsevier Butterworth-Heinemann.

29. Mulcahy, R. (2010). *Risk Management: Tricks of the Trade® for Project Managers: A course in a book*. Minneapolis, MN: RMC Publications.

30. Moen, R., & Norman, C. (2010). Clearing up Myths About the Deming Cycle and Seeing How It Keeps Evolving. Najdeno 20. junija 2014 na spletnem naslovu http://apiweb.org/circling-back.pdf

31. Moore, D. (2015). Maturity Profile Report. Najdeno 30.3.2016 na spletnem naslovu http://cmmiinstitute.com/resources/process-maturity-profile-july-2015

32. Nagrecha, S. (2002). An Introduction to Earned Value Analysis. Najdeno 1. julija 2014 na spletnem naslovu http://www.pmiglc.org/COMM/Articles/0410_nagrecha_eva-3.pdf

33. Paulk M., Curtis B., Chrissis M., & Weber C. (1993). Capability Maturity Model for Software. Version 1.1. Najdeno 13. marca 2014 na spletnem naslovu http://www.sei.cmu.edu/reports/93tr024.pdf

34. Project Management Knowledge. (2010). Residual Risk. Najdeno 23. marca 2014 na spletnem naslovu http://project-management-knowledge.com/definitions/r/residual-risk

35. PMI – Project Management Institute. (2008). *A Guide to the Project Management Body of Knowledge (PMBOK© Guide)*. Newtown Square, PA: Project Management Institute.

36. PMI – Project Management Institute. (2009). *Practice Standard for Project Risk Management*. Newtown Square, PA: Project Management Institute.

37. PRINCE2 – The Stationery Office. (2002). *Managing Successful Projects with PRINCE2*. London: Office of Government Commerce.

38. Ritchie-Dunham, J. (2013). Strategic Decision Making, Overcoming Heuristics and Biases. Najdeno 15. julija 2014 na spletnem naslovu http://instituteforstrategicclarity.org/wp-content/uploads/2013/09/S3-Heuristics-and-Biases.pdf

39. Ropponen, J., & Lyytinen, J. (2000). Components of Software Development Risk: How to Address Them? A Project Management Survey. Najdeno 26. marca 2014 na spletnem naslovu http://www-public.it-sudparis.eu/~gibson/Teaching/CSC7003/ReadingMaterial/RopponenLyytinen00.pdf

40. Royce, W. (1970). Managing the Development of Large Software Systems. Najdeno 1. junija 2014 na spletnem naslovu http://www.serena.com/docs/agile/papers/Managing-The-Development-of-Large-Software-Systems.pdf

41. Stare, A. (2010). *Obvladovanje sprememb v izvedbi projekta* (doktorska disertacija). Ljubljana: Ekonomska fakulteta.

42. Stare, A. (2011a). Planiranje virov projekta. Najdeno 28. junija 2014 na spletnem naslovu http://projektni-management.si/2011/02/19

43. Stare, A. (2011b). Kontroliranje stroškov projekta. Najdeno 28. junija 2014 na spletnem naslovu http://projektni-management.si/2011/08/29/kontroliranje-stroskov-projekta-eva-evm

44. Stare, A. (2011c). *Projektni management: teorija in praksa*. Ljubljana: Agencija Poti.

45. Standish Group. (2009). Chaos Summary 2009, The 10 Laws of Chaos. Najdeno 8. aprila 2014 na spletnem naslovu http://emphasysbrokeroffice.com/files/2013/04/Standish-Group-CHAOS-Summary-2009.pdf

46. Standish Group. (2013). Chaos Manifesto 2013, Think Big, Act Small. Najdeno 8. aprila 2014 na spletnem naslovu http://www.versionone.com/assets/img/files/ChaosManifesto2013.pdf

47. Transparency Market Research. (2013). Embedded System Market – Global Industry Analysis, Size, Share, Growth, Trends and Forecast, 2012–2018. Najdeno 16. marca 2014 na spletnem naslovu http://www.transparencymarketresearch.com/embedded-system.html

48. Wiegers, E. (1996). Misconceptions of the Capability Maturity Model. Najdeno 15. junija 2014 na spletnem naslovu http://www.processimpact.com/articles/miscon.html

PRILOGE

KAZALO PRILOG

Priloga 1: Splošna (hitra) ocena tveganosti projekta (Quick risk assesment)

Tabela 1: Vrednosti splošne (hitre) ocene tveganosti projekta po posameznih področjih

1.	**Funkcijske zahteve**	**D (%)**	**Sp (%)**	**Dt (%)**
1.1	Možne spremembe zahtev tekom projekta poznane	40,0 %	80,0 %	32,0 %
1.2	Funkcijske zahteve analizirane	10,0 %	95,0 %	9,5 %
1.3	Funkcijske zahteve dokumentirane in potrjene s strani naročnika	10,0 %	100,0 %	10,0 %
1.4	Funkcijske zahteve revidirane in potrjene s strani razvoja	10,0 %	100,0 %	10,0 %
1.5	Tehnična specifikacija izdelana in potrjena s strani razvoja	10,0 %	90,0 %	9,0 %
1.6	Funkcijske omejitve opisane in potrjene s strani razvoja in naroč.	10,0 %	95,0 %	9,5 %
1.7	Funkcijske prioritete in dovoljena odstopanja definirana in potrjena	10,0 %	80,0 %	8,0 %
		100,0 %		88,0 %
2.	**Zasnova (Design)**	**D (%)**	**Sp (%)**	**Dt (%)**
2.1	Tehnična specifikacija pokriva vse funkcijske sklope	10,0 %	95,0 %	9,5 %
2.2	Sistemska zasnova revidirana	10,0 %	100,0 %	10,0 %
2.3	Vmesniki med posameznimi komponentami definirani in revidirani	10,0 %	95,0 %	9,5 %
2.4	Izvedba modulov in uporabljenih algoritmov usklajena	10,0 %	100,0 %	10,0 %
2.5	Izvedba končnih avtomatov in sosledje faz usklajeno	10,0 %	75,0 %	7,5 %
2.6	Usklajena časovna izvedba (scheduling) modulov	10,0 %	100,0 %	10,0 %
2.7	Zahteve po resursih (spomin in CPU), potreben pretok podatkov	10,0 %	80,0 %	8,0 %
2.8	Zasnova modularna, vsi moduli izvedeni kot neodv. funk. enote	10,0 %	100,0 %	10,0 %
2.9	Razdelan management konkurenčnih procesov	10,0 %	70,0 %	7,0 %
2.10	Delež zunanjih komponent v %	10,0 %	70,0 %	7,0 %
		100,0 %		88,5 %
3.	**Razvojni proces**	**D (%)**	**Sp (%)**	**Dt (%)**
3.1	Proces je formaliziran	10,0 %	100,0 %	10,0 %
3.2	Pretekle izkušnje (lessons learned) so upoštevane	20,0 %	100,0 %	20,0 %
3.3	Management verzij je postavljen in definiran	10,0 %	100,0 %	10,0 %
3.4	Možna ponovna uporaba že izdelanih in preizkušenih komponent	10,0 %	90,0 %	9,0 %
3.5	Zahtevana kakovost dogovorjena z naročnikom in dokumentirana	20,0 %	100,0 %	20,0 %
3.6	Razvojni proces prilagojen želenemu rezultatu projekta	10,0 %	90,0 %	9,0 %
3.7	Dogovorjen razvoj le nujno potrebnih funkcij	20,0 %	50,0 %	10,0 %
		100,0 %		88,0 %
4.	**Organizacija in okolje**	**D (%)**	**Sp (%)**	**Dt (%)**
4.1	Možne spremembe v manag. projekta ali organizacije poznane	20,0 %	100,0 %	20,0 %
4.2	Poslovna politika je v soglasju s cilji projekta	35,0 %	90,0 %	31,5 %
4.3	Organizacija je stabilna	20,0 %	100,0 %	20,0 %
4.4	Zunanje poslovno okolje je stabilno	20,0 %	60,0 %	12,0 %
4.5	Možno prestrukturiranja organizacije tekom projekta je poznano	5,0 %	10,0 %	0,5 %
		100,0 %		84,0 %

se nadaljuje

nadaljevanje

5.	Zahtevnost projekta	D (%)	Sp (%)	Dt (%)
5.1	Poznana kompleksnost povezav med komponentami sistema	50 %	90 %	45,00 %
5.2	Definirana časovna sinhronizacija posameznih komponent	50 %	90 %	45,00 %
		100 %		90,00 %
6.	**Tehnologija**	**D (%)**	**Sp (%)**	**Dt (%)**
6.1	Na projektu uporabljena nova tehnologija je preizkušena	25 %	75 %	18,7 5%
6.2	Znane pomanjkljivosti še nezrele tehnologije	25 %	90 %	22,50 %
6.3	Izvedene bodo prototipne faze	25 %	100 %	25,00 %
6.4	Zunanjih moduli so preverjeni	25 %	90 %	22,50 %
		100 %		88,75 %
7.	**Planiranje in nadzor**	**D (%)**	**Sp (%)**	**Dt (%)**
7.1	Management projekta ima izkušnje iz prejšnjih projektov	30 %	100 %	30,00 %
7.2	Zagotovljen nadzor projekta	10 %	100 %	10,00 %
7.3	Planiranje potrebnih virov vključuje tveganje	20 %	90 %	18,00 %
7.4	V projekt so vključene rezerve	10 %	90 %	9,00 %
7.5	Obstaja načrt komunikacij med udeleženci projekta	10 %	10 %	1,00 %
7.6	Top-management podpira projekt in ni nasprotnikov	20 %	100 %	20,00 %
		100 %		88,00 %

Legenda: D – delež/utežna vrednost, Sp – stopnja pokritja, Dt – delež tveganja.

Priloga 2: Izhodiščna kvalitativna ocena tehničnih tveganj

Tabela 2: Izhodiščna kvalitativna ocena tehničnih tveganj

Tveganje	Element/Aktivnost	P	I	V	Ocena
F4.1.1	Prikaz grafičnih komponent	0,8	7	6	336
		0,8	8	6	384
		0,7	6	5	210
	P	0,8		M	336
Tveganje	**Element/Aktivnost**	**P**	**I**	**V**	**Ocena**
F4.1.2	Procesiranje podatkovnih paketov	0,4	3	3	36
		0,4	4	4	64
		0,6	6	5	180
	P	0,4		M	64
Tveganje	**Element/Aktivnost**	**P**	**I**	**V**	**Ocena**
F4.1.3	Ponavljanje paketov v primeru izpada	0,8	5	5	200
		0,8	6	4	192
		0,9	6	7	378
	P	0,8		M	200

se nadaljuje

nadaljevanje

Tveganje	Element/Aktivnost	P	I	V	Ocena
F4.1.4	Polnjenje in praznjenje vmesnega pomnilnika	0,2	2	1	4
		0,3	2	1	6
		0,1	1	2	2
	P	0,2		M	4
Tveganje	**Element/Aktivnost**	**P**	**I**	**V**	**Ocena**
F4.1.5	Preklop med stranmi prikazovanja	0,2	1	1	2
		0,2	1	1	2
		0,2	1	1	2
	P	0,2		M	2
Tveganje	**Element/Aktivnost**	**P**	**I**	**V**	**Ocena**
F4.1.6	Preveliko število grafičnih elementov	0,5	5	5	125
		0,4	5	4	80
		0,4	6	5	120
	P	0,4		M	120
Tveganje	**Element/Aktivnost**	**P**	**I**	**V**	**Ocena**
F4.2	Potreba po zaščiti podatkov	0,5	5	5	125
		0,4	5	4	80
		0,4	6	5	120
	P	0,4		M	120
Tveganje	**Element/Aktivnost**	**P**	**I**	**V**	**Ocena**
F4.3.1	Neznane napake/Android	0,5	4	5	100
		0,4	4	5	80
		0,4	4	5	80
	P	0,4		M	80
Tveganje	**Element/Aktivnost**	**P**	**I**	**V**	**Ocena**
F4.3.2	Neznane napake/iOS	0,6	6	5	180
		0,5	6	4	120
		0,6	6	5	180
	P	0,6		M	180
Tveganje	**Element/Aktivnost**	**P**	**I**	**V**	**Ocena**
F4.3.3	Neznane napake/Windows	0,5	5	5	125
		0,4	6	4	96
		0,6	6	5	180
	P	0,5		M	125

Legenda: P – verjetnost, I – ocena vpliva, V – ocena ranljivosti, M – srednja vrednost.

Priloga 3: Kvalitativne ocene tehničnih tveganj v različnih kontrolnih točkah

Tabela 3: Kvalitativne ocene tehničnih tveganj v različnih kontrolnih točkah

		M1				M2			
Tveg.	**Element/Aktivnost**	**P**	**I**	**V**	**R**	**P**	**I**	**V**	**R**
F4.1.1	Prikaz grafičnih komponent	0,6	7	6	252	0,4	7	6	168
		0,6	8	6	288	0,4	8	4	128
		0,6	6	5	180	0,5	6	5	150
	P	0,6		M	252	0,4		M	150
Tveg.	**Element/Aktivnost**	**P**	**I**	**V**	**R**	**P**	**I**	**V**	**R**
F4.1.3	Ponavljanje paketov	0,8	5	5	200	0,8	5	**5**	200
		0,8	6	4	192	0,8	6	**4**	192
		0,9	6	7	378	0,9	6	**7**	378
	P	0,8		M	200	0,8		M	200
Tveg.	**Element/Aktivnost**	**P**	**I**	**V**	**R**	**P**	**I**	**V**	**R**
F4.1.6	Preveliko število gr. elem.	0,3	4	5	60	0,0	4	5	0
		0,3	3	4	36	0,0	3	4	0
		0,3	4	5	60	0,0	4	5	0
	P	0,3		M	60	0,0		M	0
Tveg.	**Element/Aktivnost**	**P**	**I**	**V**	**R**	**P**	**I**	**V**	**R**
F4.2	Potreba po zaščiti podatkov	0,5	5	5	125	0,0	5	5	0
		0,4	5	4	80	0,0	5	4	0
		0,4	6	5	120	0,0	6	5	0
	P	0,4		M	120	0,0		M	0
Tveg.	**Element/Aktivnost**	**P**	**I**	**V**	**R**	**P**	**I**	**V**	**R**
F4.3.1	Neznane napake/Android	0,5	4	5	100	0,3	4	5	60
		0,4	4	5	80	0,3	4	5	60
		0,4	4	5	80	0,3	4	5	60
	P	0,4		M	80	0,3		M	60
Tveg.	**Element/Aktivnost**	**P**	**I**	**V**	**R**	**P**	**I**	**V**	**R**
F4.3.2	Neznane napake/iOS	0,6	6	5	180	0,6	6	5	180
		0,5	6	4	120	0,5	6	4	120
		0,6	6	5	180	0,6	6	5	180
	P	0,6		M	180	0,6		M	180
Tveg.	**Element/Aktivnost**	**P**	**I**	**V**	**R**	**P**	**I**	**V**	**R**
F4.3.3	Neznane napake/Windows	0,4	5	5	100	0,4	5	5	100
		0,4	5	4	80	0,4	5	4	80
		0,5	6	5	150	0,4	4	5	80
	P	0,4		M	100	0,4		M	80

se nadaljuje

nadaljevanje

		M3				M4			
Tveganje	**Element/Aktivnost**	**P**	**I**	**V**	**R**	**P**	**I**	**V**	**R**
F4.1.1	Prikaz grafičnih komponent	0,3	7	6	126	0,2	5	6	60
		0,3	8	6	144	0,2	8	6	96
		0,3	6	5	90	0,2	5	5	50
	P	0,3		M	126	0,2		M	60
Tveganje	**Element/Aktivnost**	**P**	**I**	**V**	**R**	**P**	**I**	**V**	**R**
F4.1.3	Ponavljanje paketov	0,8	5	5	200	0,2	5	5	50
		0,8	6	4	192	0,2	6	4	48
		0,9	6	7	378	0,2	6	7	84
	P	0,8		M	200	0,2		M	50
Tveganje	**Element/Aktivnost**	**P**	**I**	**V**	**R**	**P**	**I**	**V**	**R**
F4.1.6	Preveliko število graf. elem.	0,0	4	5	0	0,0	4	5	0
		0,0	3	4	0	0,0	3	4	0
		0,0	4	5	0	0,0	4	5	0
	P	0,0		M	0	0,0		M	0
Tveganje	**Element/Aktivnost**	**P**	**I**	**V**	**R**	**P**	**I**	**V**	**R**
F4.2	Potreba po zaščiti podatkov	0,0	5	5	0	0,0	5	5	0
		0,0	5	4	0	0,0	5	4	0
		0,0	6	5	0	0,0	6	5	0
	P	0,0		M	0	0,0		M	0
Tveganje	**Element/Aktivnost**	**P**	**I**	**V**	**R**	**P**	**I**	**V**	**R**
F4.3.1	Neznane napake/Android	0,2	4	5	40	0,2	2	5	20
		0,2	3	5	30	0,2	2	5	20
		0,2	3	5	30	0,4	4	2	32
	P	0,2		M	30	0,2		M	20
Tveganje	**Element/Aktivnost**	**P**	**I**	**V**	**R**	**P**	**I**	**V**	**R**
F4.3.2	Neznane napake/iOS	0,5	6	5	150	0,2	6	5	60
		0,5	5	4	100	0,2	5	4	40
		0,5	4	5	100	0,2	4	5	40
	P	0,5		M	100	0,2		M	40
Tveganje	**Element/Aktivnost**	**P**	**I**	**V**	**R**	**P**	**I**	**V**	**R**
F4.3.3	Neznane napake/Windows	0,3	5	5	75	0,2	3	5	30
		0,3	5	4	60	0,2	3	4	24
		0,3	4	5	60	0,1	3	5	15
	P	0,3		M	60	0,2		M	24

Legenda: P – verjetnost, I – ocena vpliva , V – ocena ranljivosti, M – srednja vrednost.

Priloga 4: Tedensko spremljanje projekta

Tabela 4: Tedensko spremljanje projekta

	T1	T2	T3	M1	T5	T6	T7	M2	T9	T10
BCWP (h)	154,00	146,00	146,00	145,00	152,00	148,00	152,00	140,00	140,00	136,00
ACWP (h)	160,00	160,00	160,00	154,00	192,00	192,00	192,00	172,00	132,00	132,00
CPI	0,96	0,91	0,91	0,94	0,79	0,77	0,79	0,81	1,06	1,03
BAC (EUR)	6.000,00	12.000,00	18.000,00	24.000,00	30.000,00	36.000,00	42.000,00	48.000,00	54.000,00	60.000,00
AC (EUR)	6.000,00	12.000,00	18.000,00	23.775,00	30.975,00	38.175,00	45.375,00	51.825,00	56.775,00	61.725,00
EAC (EUR)	124.675,32	131.506,85	131.506,85	127.448,28	151.578,95	155.675,68	151.578,95	147.428,57	110.880,00	114.141,18
BAC30 (EUR)	156.000,00	156.000,00	156.000,00	156.000,00	156.000,00	156.000,00	156.000,00	156.000,00	117.600,00	117.600,00
BAC30 + EWCC (EUR)	236.880,00	236.880,00	236.880,00	236.880,00	228.720,00	228.720,00	228.720,00	228.720,00	162.720,00	162.720,00
	T11	**M3**	**T13**	**T14**	**T15**	**M4**	**T17**	**T18**	**T19**	**M5**
BCWP (h)	156,00	160,00	160,00	145,00	142,00	140,00	145,00	145,00	140,00	132,00
ACWP (h)	132,00	132,00	132,00	132,00	132,00	132,00	132,00	132,00	132,00	132,00
CPI	1,18	1,21	1,21	1,10	1,08	1,06	1,10	1,10	1,06	1,00
BAC (EUR)	66.000,00	72.000,00	78.000,00	84.000,00	90.000,00	96.000,00	102.000,00	108.000,00	114.000,00	120.000,00
AC (EUR)	66.675,00	71.625,00	76.575,00	81.525,00	86.475,00	91.425,00	96.375,00	101.325,00	106.275,00	111.225,00
EAC (EUR)	99.507,69	97.020,00	97.020,00	107.056,55	109.318,31	110.880,00	107.056,55	107.056,55	110.880,00	117.600,00
BAC30 (EUR)	117.600,00	117.600,00	117.600,00	117.600,00	117.600,00	117.600,00	117.600,00	117.600,00	117.600,00	117.600,00
BAC30 + EWCC (EUR)	162.720,00	162.720,00	158.040,00	158.040,00	158.040,00	158.040,00	138.000,00	138.000,00	138.000,00	138.000,00

Legenda:

T1 .. T19 – Zaporedna številka tedna, M1 .. M5 – zaporedna številka tedna z vmesnim ciljem,
BCWP – planirani obseg ur, ACWP – opravljene ure, CPI – indeks stroškovne učinkovitosti,
BAC – planirani strošek dela, AC – dejanski strošek dela, EAC – napoved stroškov ob zaključku projekta,
BAC30 – planirani stroški z dodano 30 % projektno rezervo,
BAC30 + EWCC – planirani stroški s projektno rezervo, povišani za skupni strošek tveganj z najslabšim možnim izidom.