UNIVERZA V LJUBLJANI

EKONOMSKA FAKULTETA

DIPLOMSKO DELO

DANA PETRIC

UNIVERZA V LJUBLJANI

EKONOMSKA FAKULTETA

DIPLOMSKO DELO

ANALIZA KOMBINIRANEGA TRADICIONALNEGA IN AGILNEGA PROJEKTNEGA PRISTOPA PRI TEHNIČNI NADGRADNJI PROGRAMSKE OPREME

Ljubljana, maj 2016 DANA PETRIC

IZJAVA O AVTORSTVU

Spodaj podpisana Dana Petric, študentka Ekonomske fakultete Univerze v Ljubljani, izjavljam, da sem avtorica diplomskega dela z naslovom Analiza kombiniranega tradicionalnega in agilnega projektnega pristopa pri tehnični nadgradnji programske opreme, pripravljenega v sodelovanju s svetovalcem doc. dr. Aljažem Staretom.

Izrecno izjavljam, da v skladu z določili Zakona o avtorski in sorodnih pravicah (Ur. l. RS, št. 21/1995 s spremembami) dovolim objavo diplomskega 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
    - poskrbela, da so dela in mnenja drugih avtorjev oziroma avtoric, ki jih uporabljam v diplomskem delu, citirana oziroma navedena v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani, in
    - pridobila 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 zapisala;
- se zavedam, da je plagiatorstvo – predstavljanje tujih del (v pisni ali grafični obliki) kot mojih lastnih – kaznivo po Kazenskem zakoniku (Ur. l. RS, št. 55/2008 s spremembami);
- se zavedam posledic, ki bi jih na osnovi predloženega diplomskega dela dokazano plagiatorstvo lahko predstavljalo za moj status na Ekonomski fakulteti Univerze v Ljubljani v skladu z relevantnim pravilnikom.

V Ljubljani, dne 04.05.2016 Podpis avtorice_____

KAZALO

KAZALO SLIK

KAZALO TABEL

UVOD

Oddelki informacijske tehnologije v podjetjih se v zadnjih letih soočajo z zelo hitrim napredkom informacijske tehnologije. Zaposleni v teh oddelki se srečujejo s povečanim obsegom zahtev uporabnikov informacijske tehnologije, saj se le – ta vsakodnevno spreminja. Nasploh pri razvijanju nove programske opreme so zahteve uporabnikov porasle. Ne samo da uporabniki in naročniki želijo visoko zanesljivost pri uporabi, želijo tudi hiter razvoj.

V diplomskem delu proučujem management projektov, z namenom izboljšanja izvedbe IT projektov v izbranem podjetju. Raziskave so namreč pokazale, da je statistično gledano 25% srednje velikih IT projektov neuspešnih. Uspešnost projekta pada z velikostjo. Po analizi iz leta 2012 se kaže, da je neuspešnost velikih projektov za 50% višja kot neuspešnost majhnih projektov (Mieritz, 2012).

Osredotočila sem se na management projektov razvoja programske opreme za podporo poslovnim procesom. Ker je projektni management zelo obširno področje, sem se osredotočila le na opredelitev zahtev (specifikacija), časovno planiranje in kadrovanje, izvedbo, udeležence in njihove vloge, načrtovanje programske opreme (izdelka), reševanje težav, obravnavo sprememb, testiranje, implementacijo in začetek uporabe. Analizirala sem tri različne projektne pristope in izpostavila razlike med njimi na prej omenjenih področjih. Cilj diplomskega dela je bil analiza uspešnosti omenjenih treh pristopov, izpostavitev težav in napak pri delu ter predlogi za delo na podobnih projektih vnaprej.

Najprej sem pojme projektnega managementa, ki jih uporabljam, opredelila natančneje. S tem želim približati temo tudi bralcu, ki ima manj izkušenj s projektnim managementom. Natančneje opisujem faze projekta in naloge projektnega managerja.

Opisala sem različne projektne pristope uporabljene pri razvoju programske opreme, ki so kombinacija dveh strokovnih področij, saj bom v nadaljevanju analizirala projekt nadgradnje programske opreme v mednarodnem podjetju. Izpostavila sem razliko med slapovnim in agilnim pristopom, z namenom razumljivega in nedvoumnega prikaza kombiniranega pristopa pri opisanem projektu. Vključila sem tudi mnenja za izbiro slapovnega ali agilnega pristopa. Schwaber in Sutherland (2012, str. 5) navajata, da so projekti razvoja programske opreme ob uporabi agilne metode trikrat uspešnejši kot projekti ob uporabi slapovne metode.

V nadaljevanju sem analizirala projekt nadgradnje programske opreme za podporo poslovnim procesom na finančnem področju v mednarodnem podjetju, ki se je izvajal v letu 2012. Ukvarjam se z analizo razumevanja projekta kot ga opisuje Wysocki (2006, str. 420), ob se ob tem osredotočam tudi na analizo težav, s katerimi smo se soočili.

V zaključnem poglavju povzamem analiziran projekt in izpostavim informacije, ki so potrebne za pravilno izbiro uporabljenega pristopa pri projektnem managementu.

1 OPREDELITEV PROJEKTA

Ker bom v kasnejših poglavjih uporabljala pojme projekt, faze projekta in projektni management, jih bom najprej opredelila.

1.1 Projekt

Slovar slovenskega knjižnega jezika (Spletna aplikacija, b.l.) opredeljuje projekt (projekt, b.l.) kot:

- kar določa, kaj se misli narediti in kako naj se to uresniči, načrt: izdelati, predložiti projekt; projekt za modernizacijo podjetja / finančni, investicijski projekt; idejni, tehnični projekt stroja; raziskovalni projekt; vesoljski projekt,
- v gradbeništvu: skupek načrtov, tehničnih opisov in popis stroškov za kak objekt, področje: arhitekti so izdelali več projektov; projekt ceste, spomenika, stavbe; projekt električne napeljave / gradbeni projekt; arhitektura: glavni projekt, glavni načrt, izdelovanje, uresničevanje tega: financirati, kreditirati projekt; sodelovati pri projektu,
- publicistično: navadno s prilastkom umetniško delo glede na namen, da se izvede, uresniči: komisija je filmski projekt potrdila; glasbeni, gledališki projekt; uspeh mladega igralca pri projektu,
- knjižno: osnutek, predlog določenega besedila: razpravljati o projektu resolucije, zakona.

V mojem diplomskem delu se bom osredotočila na uporabo besede »projekt« iz prve alineje in sicer predvsem na projekte v informacijski tehnologiji.

Na spletu v islovarju (Poslovna aplikacija, b.l.) sem našla tudi enostavno in kratko opredelitev projekta (projekt, b.l.):

- Projekt je ciljno usmerjen in zaključen proces razvijanja dejavnosti, ki so usmerjene k doseganju končnega cilja.

V Guide to the Project Management Body of Knowledge, (v nadaljevanju PMBOK), (Guide to the Project Management Body of Knowledge, 2008, str. 5) je projekt opredeljen kot:

- Začasno prizadevanje podjetja za izdelavo edinstvenega izdelka (lahko del drugega izdelka ali pa samostojen izdelek), storitve (zmožnost opraviti storitev) ali rezultata (raziskava, katere zaključek je možno uporabiti).

Kerzner (2009, str. 2) meni, da je projekt kakršnakoli serija aktivnosti in nalog ki:

- ima določen cilj, ki mora biti izpolnjen ob določenih zahtevah,
- ima določen začetek in konec,
- ima investicijske limite,

- uporablja človeške in nečloveške vire (denar, delovno silo, opremo).

Projekt je torej delo, kjer je cilj dogovorjen in določen. Opredeljeni so tudi viri in sredstva za dosego tega cilja, kot tudi časovni, kadrovski in stroškovni okviri. V projektu so viri za izvedbo projekta v naprej določeni in omejeni in kot taki, tudi povezani s tveganjem. Rezultat uspešnega projekta je vedno nekaj novega, lahko je izdelek ali storitev.

1.2 Faze projekta

Strokovnjaki različno opisujejo faze projekta, ki jih je Stare (2011, str. 20) povzel in opredelil v štiri najbolj očitne in logične faze, kot prikazuje slika 1.

Snovanje je faza projekta, kjer opredelimo idejo oz. potrebo. Preučimo možnost izvedbe projekta in ocenimo vire.

Priprava je faza projekta, kjer se izdela plan projekta, ki vsebuje seznam zahtev, seznam izvajalcev projekta, plan stroškov in plan obvladovanj tveganj, kot tudi sistem komuniciranja med udeleženci projekta.

Izvedba je faza projekta, ki vsebuje izvajanje planiranih aktivnosti kot je bilo opredeljeno v fazi planiranja. Pomembno je usklajevanje vseh udeležencev, vodenje tima in kontroliranje izvedbe

Zaključek je zaključna faza projekta, kjer preverimo izdelek, ga predamo naročniku in formalno zaključimo projekt z zaključnim poročilom. Vključimo lahko tudi pridobljena spoznanja (angl. *lessons learned*), ki bi lahko pripomogla k boljši izvedbi prihodnjih projektov.

Slika 1: Faze projekta

Vir: A. Stare, Projektni management: teorija in praksa, 2011, str. 21.

Odločitev o snovanju vključuje idejo in v naslednjem koraku preučitev naročnika, da ideja prinaša koristi podjetju.

Odločitev o pripravi vključuje potrditev naročnika, da se pripravi plan izvedbe oz. specifikacija zahtev za projekt.

Odločitev o izvedbi pa je potrdilo in odobritev za izvajanje projekta po pripravljenem planu.

Naročnik je običajno uporabnik novo razvite programske opreme. Projektni tim je skupina ljudi, ki je odgovorna, da bo izdelek izveden na dogovorjen način.

Na sliki 1 je koriščenje svetlejše orisano, ker je to področje, ki ne spada več med faze projekta, je pa pravzaprav cilj oz. izdelek projekta. To, da naročnik na koncu v realnosti uporablja oz. koristi narejeno je faza koriščenja.

1.3 Projektni management

Na spletu v islovarju (Poslovna aplikacija, b.l.) sem našla enostavno in kratko opredelitev projektnega managementa (projektni management, b.l.):

- Je planiranje, organiziranje in kontroliranje projektov v organizaciji

PMBOK (2008, str. 6) opredeli projektni management kot uporabo znanja, veščin orodij in tehnik pri aktivnostih v projektu za zadovoljevanje zahtev projekta. Projektni manager skrbi za zagon aktivnosti, koordinira in vodi tim ipd. Vse to izvaja s primerno uporabo in s povezovanjem dvainštirideset logično povezanih procesov projektnega managementa razdeljenih po prej navedenih fazah.

Projektni management je skupek znanj in veščin za obvladovanje planiranja, urnika, stroškov, obsega,tveganja, človeških virov in udeležencev v projektu. O tem Project Management Institute (v nadaljevanju PMI) govori v področjih znanja (angl. *knowlege areas)*; (Brandon, 2006, str.35).

Področja znanja projektnega managementa po PMBOK, ki jih uporablja tudi Houston metoda projektnega managementa so:

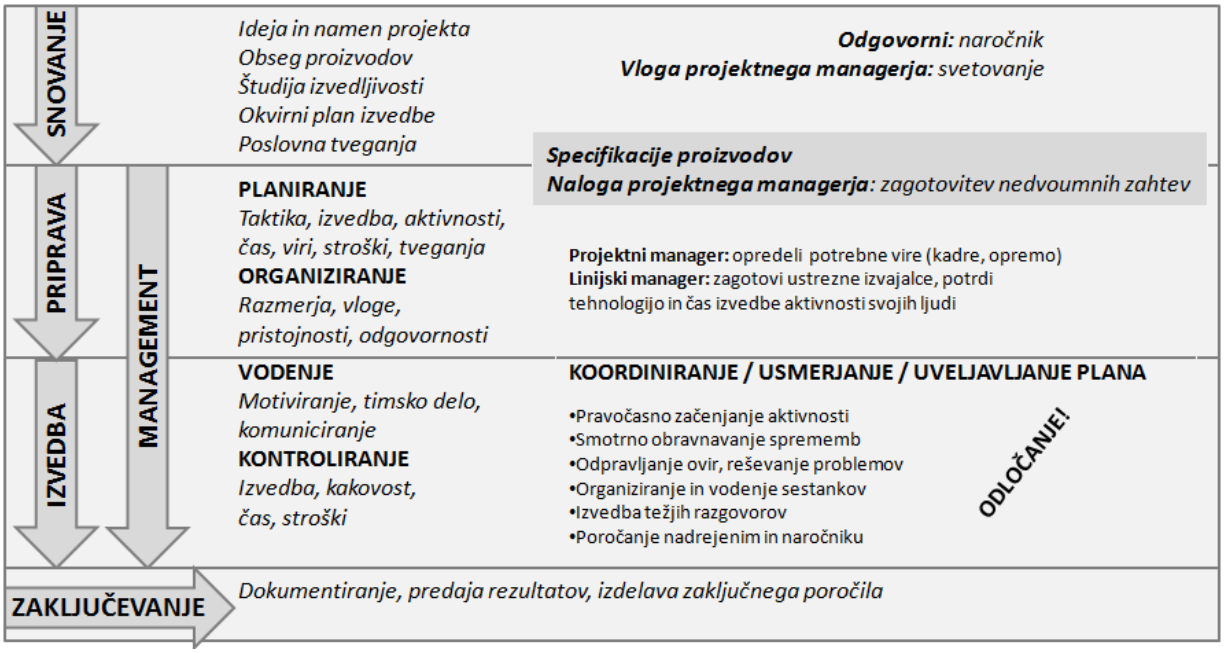
1. Management integracije projekta (angl. *Integration management*),
2. Management obsega projekta (angl. *Scope management*),
3. Management časa projekta (angl. *Time management*),
4. Management stroškov projekta (angl. *Cost management*),
5. Management kakovosti projekta (angl. *Quality management*),
6. Management človeških virov v projektu (angl. *Human Resource management*),
7. Management komuniciranja v projektu (angl. *Communication management*),
8. Management tveganj projekta (angl. *Risk management*),
9. Management oskrbovanja projekta (angl. *Procurement management*).

Projektni management se opredeljuje kot veščino vodenja ljudi ter koordiniranja dela skozi celoten življenjski ciklus projekta, kot prikazujem na sliki 2. Projektni manager z uporabo sodobnih vodstvenih tehnik želi doseči planirane strateške in operativne cilje ter zadovoljiti

potrebe in pričakovanja udeležencev projekta ob upoštevanju planiranega obsega, stroškov, kakovosti in časa.

Na splošno bi lahko povzela (Slika 2), da je projektni manager aktiven v fazi snovanja projekta kot svetovalec za zagotovitev nedvoumnih zahtev, v fazi planiranja projekta izbira taktiko za izvedbo, zagotavlja potrebne vire, v fazi izvedbe spremlja izvajanje, motivira, rešuje probleme, kontrolira stroške, časovne okvire in tveganja, v fazi zaključevanja organizira dokumentiranje in izdelavo zaključnega poročila.

Slika 2: Naloge projektnega managerja

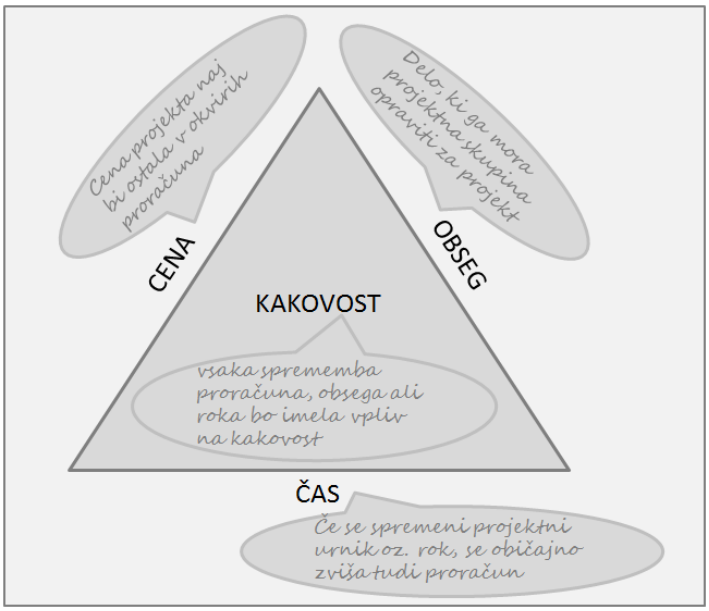


Vir: Stare, Vloga in naloge managerja projekta, 2010.

Projektni manager je omejen z določenimi zahtevami naročnika projekta. Nekateri jih imenujejo cilji projektnega managementa, drugi omejitve projekta, najpogosteje pa jih imenujejo železni trikotnik (angl. *Project Management Triangle*). Obstaja mnogo različic železnega trikotnika. Na sliki 3 predstavljam eno izmed njih.

Železni trikotnik prikazuje omejenost in povezanost kakovosti z obsegom, proračunom in časom. Projektni manager naj upošteva dejstva železnega trikotnika v vseh fazah projekta. Vsaka izmed stranic trikotnika predstavlja omejitev in ob spremembi ene omejitve, se spremenita tudi preostali dve omejitvi (Rowley, 2013).

Slika 3: Železni trikotnik



Vir: Stellman, A Few Things Every Job-Seeking Programmer Should Know About Project Management (part2) 2010, & Green & Stellman, HeadFirstPMP, 2007.

2 RAZLIČNI PROJEKTNI PRISTOPI RAZVOJA PROGRAMSKE OPREME

V nalogi bom izpostavila le tri izmed mnogih projektnih pristopov, ki so dejansko kombinacija dveh strokovnih področij – projektnega managementa in pristopa razvoja programske opreme. To so zaporedni, agilni in Houston pristop. Za to sem se odločila, ker smo v podjetju uporabili kombinacijo teh treh in ker želim analizirati, kje smo sledili teoriji in se je izkazalo za primerno oz. neprimerno in kje smo zanemarili teorijo in kakšen je bil rezultat tega.

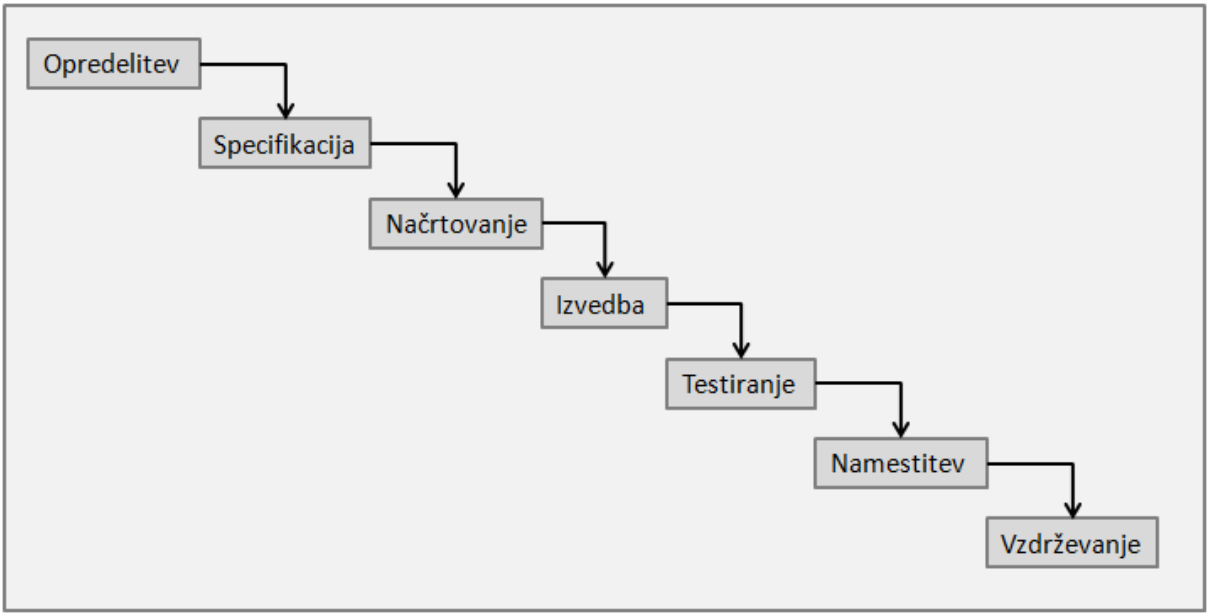
2.1 Zaporedni (angl. *waterfall*) pristop

Ker se izrazi slapovni, slapast, zaporedni ali model vodnega slapa uporabljajo šele odkar so se pojavili novejši projektni pristopi – terminologija v Sloveniji še ni enotna. Uporabljala bom izraz slapovni pristop (angl. *waterfall*).

Najpogostejša in uveljavljena metodologija uporabljena pri izgradnji IT sistemov je bila neformalno imenovana kot »slapovna« metodologija, oz. formalno imenovana kot »metodologija življenjskega cikla razvoja programske opreme« (angl. *Software development life cycle methodology*, v nadaljevanju SDLC). Brandon navaja, da je ta pojem in izraz prvič uporabil Royce (1970) pri informacijski tehnologiji (Brandon, 2006, str 73).

Faze pri tej klasični metodologiji so prikazane na sliki 4. V teoriji naj se te faze ne prekrivajo ali ponavljajo, kot je razvidno na sliki 4 (Brandon, 2006, str. 73).

Slika 4: Metodologija življenjskega cikla razvoja programske opreme



Vir: D. Brandon, Project Management for Modern Information Systems, 2006, str. 73.

Fazi opredelitve in specifikacije pri tej metodologiji sta skladni z opredelitvijo faze snovanja pri projektih. Načrtovanje programske opreme je skladno s fazo planiranja izvedbe del. In tu ugotovim, da se SDLC ne ukvarja z organizacijskim vidikom projekta ampak le s tehnologijo izvedbe. Pristop zanemarja omejitve projekta oz. predpostavlja, da se s tem ukvarja projektni manager. Izvedba, testiranje sistema in integracije in sama namestitev programske opreme je skladna s projektno fazo izvedbe. Testiranje, namestitev in vzdrževanje je skladno s fazo zaključevanja projekta.

Faza opredelitve vključuje jasen opis ciljev in ugotavljanje kako in zakaj bo nova programska oprema boljša, cenejša in hitrejša od obstoječe opreme ali ročnega dela, ki ga bo nadomeščala. Običajno vsebuje grobo analizo stroškov in koristi. Za to fazo je značilno pogosto pogajanje in dogovarjanje s kupcem o času, stroških in projektnem planu, ki se običajno sklene z zeleno lučjo za projekt in podpisano projektno listino oz. naročilo projekta. Projektna listina vsebuje idejo oz. namen projekta, merljive cilje, obseg z navedbo kaj bo narejeno in kaj ne, vključuje glavne deležnike z imeni, če pa še niso znani vključuje vlogo deležnika, opisani so mejniki in proračun projekta, vsebuje znane omenjitve projekta, obstoječa pomembna dejstva, tveganja in odvisnosti z drugimi projekti ali procesi, planiran proračun in časovne roke.

Faza specifikacije vključuje opis vseh zahtev z uporabo primerov uporabe (angl. use case), pripravo predhodnega priročnika, podroben plan projekta in tehnično strukturo projekta (angl. *WS – work breakdown structure*). Specificirati je potrebno tudi oceno časa

in proračuna in pripraviti analizo stroškov in koristi (angl. *cost benefit*). Vse skupaj odobreno in potrjeno s strani deležnikov projekta. Razlog za pripravo osnutka priročnika za uporabo je, da se lahko na osnovi le-tega začne planiranje testiranja in planiranje usposabljanja, ter planiranje morebitnih drugih nalog, ki so odvisne od le-tega.

Faza načrtovanja programske opreme vključuje reševanje kritičnih vprašanj, izbiro arhitekture in računalniškega okolja, sprejetje standardov, razporeditev človeških virov, oblikovanje podatkovne in strukturne baze, načrtovanje algoritmov in procesov, priprava matrike zahtevkov, načrt za predhodno testiranje, finalni popravki ocene časovnih rokov in proračuna, kot tudi analiza stroškov in koristi. Pogosto je korak načrtovanja razdeljen na analizo in na oblikovanje podrobnega načrta (angl. *design*) programske opreme.

Faza izvedbe vključuje implementacijo podrobnega načrta: npr. kodiranje, testiranje modulov, sistemsko integracijo, pripravo osnutka interne dokumentacije in izvedba načrtovanega testa spiska programskih navodil.

Faza testiranja vključuje celostno integracijo programske opreme in gradiv, ki obsega sistem testiranja, dokončanje dokumentacije za uporabnika, pripravo učnega gradiva, pripravo navodil za namestitev in priročnika za izvedbo namestitve programske opreme (angl. *roll-out*).

Faza namestitve vključuje izvedbo namestitve programske opreme, usposabljanje končnih uporabnikov, pripravo priročnika z izkušnjami iz projekta (angl. *lessons learned*), določitev postopkov za rokovanje, podporo uporabnikom in upravljanje z nastavitvami programske opreme (angl. *configuration management*).

Faza vzdrževanja vključuje upoštevanje vseh vnaprej določenih postopkov za reševanje težav s programsko opremo, upoštevanje komunikacijskih poti v primeru sporov. Vključuje tudi postopke varnostnega kopiranja (angl. *backup*), zaščite pred vdori, kontrolo zmogljivosti in kvalitete programske opreme (Brandon, 2006, str. 75-76).

Na koncu vsake faze je potrebno izvesti formalno srečanje na katerem se pripravi dokument s katerim udeleženci projekta potrdijo zaključek pretekle faze. V teoriji naj bi bil rezultat takega postopka ob zaključku vsake faze izdelek oz. programska oprema, ki zadovoljuje začetne zahteve kot tudi udeležence projekta

Mnogi avtorji s področja managementa IT projektov menijo, da je klasičen slapovni pristop prepočasen v današnjem hitrem tempu in hitro spreminjajočem svetu (Brandon, 2006, str. 76). Menim, da je klasičen slapovni pristop primeren za projekte, ki morajo biti izdelani naenkrat v celoti. Se pa strinjam, da je današnjemu hitremu tempu življenja in internetnim aplikacijam potrebno zagotoviti hitro reakcijo na potrebe uporabnikov in naročnikov. Čeprav včasih naročnik v kratkem času dobi le del uporabnih funkcionalnosti. Kot primer bi navedla izdelavo spletne strani, kjer je lahko vsaka stran izdelana kot zaključena delujoča celota. Med drugim May in Zimmer že leta 1996 navajata, da vedno več razvojnih timov programske opreme uporablja manjše objave, hitrejše dobave

uporabnikom in dinamične procese in planiranje, kar poimenujeta metodo evolucijskega razvoja (angl. *evolutionary development*) (May & Zimmer, 1996).

Brandon omenja tudi druge metodologije, zato bi jih tu le na kratko omenila: Overlap method, Evolutionary development, Incremental development or Bounding box development. Skupno tem metodam je, da želijo pohitriti razvoj in prepletajo različne faze. Vsaka od metod ima prednosti in slabosti in jo je priporočljivo uporabljati glede na naravo, tip in velikost IT projekta.

2.2 Agilni pristop

Osnovno izhodišče nove šole mišljenja je povezano s pojmom agilnosti. V slovenščini temu pravimo gibčnost ali prilagodljivost.

Brandon navaja, da je agilno programiranje ime za vse večje število lahkih metodologij kot na primer Crystal (Cockburn, 2001), Scrum (Schwaber, 2001), Ekstremno programiranje (Beck, 1999)…. V devetdesetih letih prejšnjega stoletja je bila velika potreba po izgradnji novih IT sistemov, kot na primer spletne aplikacije in e-trgovine, obravnavati je bilo potrebno tudi Y2K (prehod na novo tisočletje, angl. *year two thousand*) problematiko. Te gibke metodologije so odpravile veliko postavk slapovne metodologije, ki so upočasnjevale razvijalce, kot so na primer podrobne opredelitve formalnih zahtev in obsežne dokumentacije (Brandon, 2006, str.88).

Leta 2001 se je v Združenih državah Amerike skupina strokovnjakov s področja lahkih metodologij sestala z namenom ugotoviti, kaj imajo njihove metodologije skupnega in na podlagi teh ugotovitev poenotiti osnove metodologij. Nastal je **Agilni manifest** (angl. *Agile Manifesto*), ki je listina za agilni razvoj programske opreme, ki določa temeljna načela in priporočila agilnih metodologij

2.2.1 Citat manifesta:

Odkrivamo boljše načine razvoja programske opreme, tako, da jo razvijamo, in pri tem pomagamo tudi drugim. Naše vrednote so ob tem postale:

- Posamezniki in sodelovanje pred procesi in orodji.
- Delujoča programska oprema pred vseobsežno dokumentacijo.
- Sodelovanje z naročnikom pred pogodbenimi pogajanji.
- Prožen odziv na spremembe pred togim, slepim sledenjem začrtanemu planu

Z drugimi besedami, četudi cenimo dejavnike na desni, vseeno bolj cenimo tiste na levi (Beck et al., 2001).

Principi v ozadju agilnega manifesta so:

1. Najvišja prioriteta agilnih managerjev je zadovoljiti stranko z zgodnjim in nepretrganim izdajanjem uporabne programske opreme.

2. Agilni managerji sprejemajo spremembe zahtev, celo v poznih fazah razvoja. Agilni procesi vprežejo tovrstne spremembe v prid konkurenčnosti stranke za katero delajo.
3. Delujočo programsko opremo agilni managerji izdajajo pogosto, znotraj obdobja nekaj tednov, do nekaj mesecev. Čim pogosteje tem bolje.
4. Uporabniki in razvijalci morajo skozi celoten projekt dnevno sodelovati.
5. Projekte gradimo okrog motiviranih posameznikov. Omogočimo jim delovno okolje, nudimo podporo in jim zaupamo, da bodo svoje delo opravili.
6. Najboljša in najučinkovitejša metoda posredovanja informacij razvojni ekipi in znotraj ekipe same, je pogovor iz oči v oči.
7. Delujoča programska oprema je primarno merilo napredka.
8. Agilni procesi promovirajo trajnostni razvoj. Skrbnik projekta, razvijalci in uporabniki morajo biti zmožni konstantnega tempa za nedoločen čas.
9. Nenehna težnja k tehnični odličnosti in k dobremu načrtovanju izboljša agilnost.
10. Preprostost -- umetnost zmanjševanja količine nepotrebnega dela -- je bistvena.
11. Najboljše arhitekture, zahteve in načrti izhajajo iz tistih ekip, ki se samostojno organizirajo.
12. V rednih časovnih razdobjih ekipa išče načine, kako postati učinkovitejša ob rednem prilagajanju svojega delovanja.

Čeprav se različne agilne metode razlikujejo, imajo nekaj skupnih stvari. Večina uporablja postopen pristop. Skupni namen je, da si okreten in lahko sprejmeš spremembe in razvijaš programsko opremo, ki je prilagodljiva. Skupna lastnost je tudi pogost kontakt z uporabniki in osredotočenost na ljudi in ne na procese, kot tudi poudarek na izgradnji skupine (angl. *team building*). Temeljna načela agilnega pristopa so (Brandon, 2006, str. 89):

- Uporabi preprost načrt.
- Načrtuj sproti in preurejaj kodo.
- Delaj male, postopne korake (ob spreminjanju ali dodajanju kode, uporabi najmanjši korak in takoj narejeno testiraj).
- Drži se ene naloge naenkrat.
- Uporabljaj IDE (angl. *integrated development environment*) in RAD (angl. *rapid application development*) orodja.
- Uporabljaj le tehnike, ki jih res obvladaš.

Prednosti, ki jih pridobiš s takimi načeli je hitrejša odzivnost na spremembe v zahtevkih, preprostost celotnega razvoja, možnost predhodnega kodiranja. Pri preurejanju kode pridobi najpomembnejši del največ pozornosti, saj ni izgube časa s preurejanjem kode, ki je ni potrebno spreminjati. Velja tudi, da je koda v teku vedno stabilna. (Brandon 2006, str. 89)

Že vrsto let vodilni v procesih priporočajo postopen in ponavljajoč način razvoja programske opreme, ki zagotavlja celo večjo funkcionalnost. Čeprav je postopen in

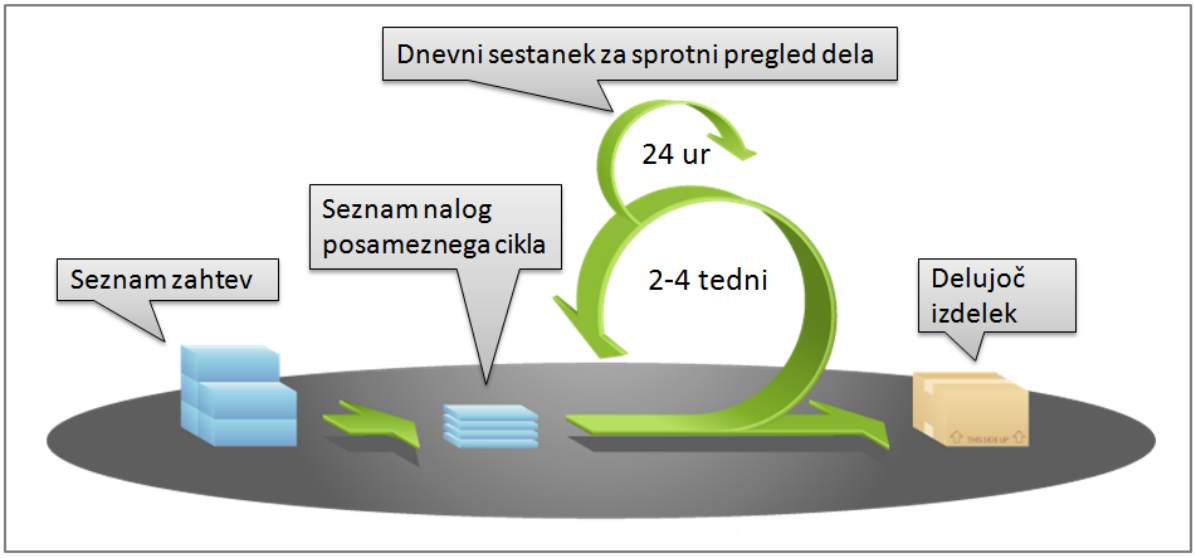
ponavljajoč način značilnost vseh agilnih projektov, se zdi, da je le-ta način manj pogost pri razvoju programske opreme (Highsmith, 2002, str. 34)

2.2.2 Opis metodologije SCRUM

Scrum je najbolj razširjena agilna metoda za razvoj programske opreme. Scrum je v bistvu ogrodje , ki ga je enostavno razumeti, vendar se ga je težko držati. Scrum namreč izbiro, na kakšen način je potrebno kaj narediti, prepušča razvojnemu timu. Razvoj po Scrum metodi poteka v časovnih intervalih, ki se imenujejo cikel ali iteracija. Prva knjiga o Scrumu je bila z naslovom *Agile software Development with Scrum* izdana leta 2001.

Pri Scrumu naletimo na drugačno izrazoslovje (Slika 5), kot se uporablja pri slapovnem pristopu.

Slika 5: Grafični prikaz enega cikla



Vir: M.Cohn, Scrum Iteration Model, 2014.

Cikel ali iteracija (angl. *sprint* ali *iteration*) je časovno omejen na 2-4 tedne in vedno traja enako dolgo. Vsak cikel je kot mali projekt, v katerem pripravimo verzijo produkta z dodano novo uporabno funkcionalnostjo. Cikel v trajanju razvoja vedno traja enako dolgo. Takoj po zaključku enega se začne naslednji cikel. V enem ciklu se odvijejo vse faze od planiranja, izvedbe do testiranja in zaključka.

Seznam zahtev (angl. *product backlog*) je urejen seznam vseh funkcionalnosti, ki jih še nameravamo dodati končnemu izdelku. Le-te so zapisane v obliki uporabniške zgodbe. Seznam zahtev je dinamičen seznam, ki se v času izvedbe projekta ves čas spreminja. Zahteve se brišejo, dodajajo in spreminjajo in sicer glede na odločitve skrbnika izdelka in glede na prednostno stopnjo. Za seznam zahtev je zadolžen skrbnik izdelka. Posamezen vnos v seznam zahtev vključuje vsebino, ki ima za skrbnika izdelka uporabno vrednost, prednostne stopnje, ter ocene, kako nujna je realizacija uporabniške zgodbe.

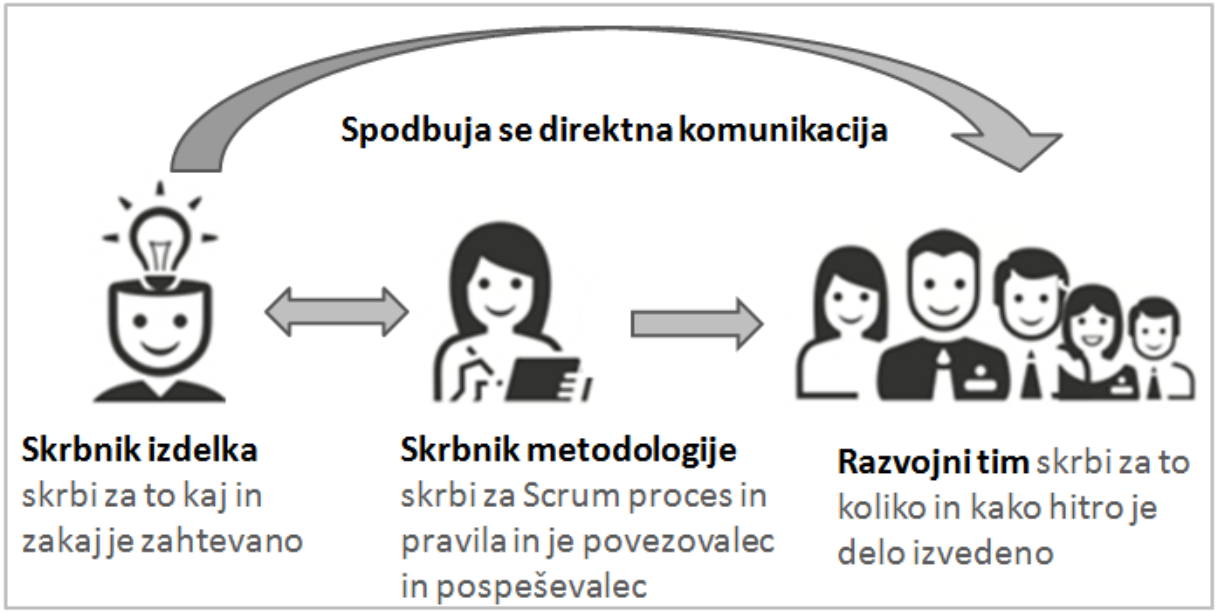
Seznam nalog (angl. *sprint backlog*) je določeno število uporabniških zgodb, ki jih izbere razvojni tim za en cikel s seznama zahtev in sicer glede na prednostno stopnjo, ki jo je določil skrbnik izdelka. V seznam nalog se uporabniške zahteve uvrstijo na sestanku za izdelavo plana cikla, kjer sodeluje tudi skrbnik izdelka. Spreminjanje seznama nalog je dovoljeno le s strani razvojnega tima.

Uporabniška zgodba (angl. *user story*) je kratek in enostaven opis želene funkcionalnosti napisana s strani uporabnika le-te. Zgodbe so običajno v takšni obliki: Kot <*tip uporabnika*>, hočem <*nek cilj*>, da <*nek razlog*>.

Dnevni sestanek za sprotni pregled dela (angl. *daily scrum*) je dnevni 15 minutni sestanek razvojnega tima za izdelavo plana za naslednjih 24 ur. Poteka vsak dan ob istem času na istem kraju . Lahko se ga udeležijo vsi udeleženci projekta, obvezen je za celoten razvojni tim, skrbnika metodologije in skrbnika izdelka.

Scrum vloge so tri (slika 6) in sicer skrbnik metodologije, skrbnik izdelka in razvojni tim.

Slika 6: Scrum vloge



Vir: S. Viscardi, The Professional ScrumMaster's Handbook, 2013.

Skrbnik metodologije (angl. *Srum Master*) ali Scrum mojster skrbi za to, da Scrum proces poteka po ustaljenem planu in da se dosledno upoštevajo pravila Scruma. Pomaga razvojnemu timu, da je produktiven, skrbi, da lahko razvojni tim nemoteno razvija in ga brani pred vsemi motečimi dejavniki. Poskrbi za odstranjevanje ovir in rešuje probleme tako, da ne obremenjuje razvojnega tima.

Skrbnik izdelka (angl. *Product Owner*) postavi temeljne zahteve projekta. Odgovoren je za aktualiziranje seznama zahtev in določitev prednostnih uporabniških zgodb seznama

nalog za en cikel. Skrbnik izdelka mora poznati tematiko zahtev in njegove odločitve morajo biti spoštovane. Lahko obstaja skupina, ki mu svetuje, a sam mora sprejeti odločitve in prioritete za seznam nalog, ki jih morajo sprejeti in upoštevati vsi.

Razvojni tim (angl. *Scrum Development Team*) skrbi za razvoj funkcionalnosti po seznamu nalog v vsakem ciklu sproti. Po vsakem ciklu mora izdati novo verzijo izdelka, ki mora biti uporabna. V razvojnem timu so vsi člani enakopravni. Sestavljena mora biti tako, da imajo skupaj vsa znanja, ki so potrebna za dosego cilja. Vsi so soodgovorni za uspeh in rezultat cikla in na koncu projekta. Scrum prepoveduje kakršnokoli spreminjaje razvojnega tima med ciklom in tudi po ciklu se odsvetuje le-to. Priporoča se od pet do devet članov, saj se je v praksi pokazalo, da je za dobro delo potrebno preveč koordinacije v timu (Schwaber & Sutherland, 2013).

Za izbiro tradicionalnega ali agilnega pristopa je potrebno razumeti projekt, stranko, poslovno okolje, podjetje in kvaliteto virov. Šele nato lahko sprejmemo odločitev kateri pristop za razvoj programske opreme je primeren. (Wysocki, 2006, stran 273)

Agilni pristop razvoja programske opreme se po mnenju Gladkove (2012) uporabi kadar:

- je potrebno dati programsko opremo v uporabo v zelo kratkem času,
- podrobnosti projekta še niso v celoti določene in se lahko v fazi razvoja programske opreme še močno spremenijo,
- želijo uporabniki osnovne oz. delne funkcionalnosti še preden bo celoten projekt zaključen,
- želijo uporabniki z razlago zahtev sodelovati in kontrolirati v celotnem razvojnem procesu programske opreme,
- je možno zagotoviti stalno prisotnost in vključevanje končnih uporabnikov programske opreme z razvijalci,
- je možno zagotoviti sprotno preverjanje razvitih funkcionalnosti s strani končnih uporabnikov.

Slapovni pristop razvoja programske opreme se uporabi kadar:

- ni potrebe po kratkoročni uporabi nove programske opreme,
- ni ovir, da bo programska oprema izgotovljena po daljšem obdobju,
- lahko sprejmete odločitev, da prejmete izpopolnjeno programsko opremo šele po daljšem obdobju,
- je projekt pred začetkom izvedbe natančno opredeljen in definiran do vseh podrobnosti,
- je pomembno, da imaš celotno dokumentacijo že na začetku projekta – v primeru zakonskih odobritev ipd.,
- je pomembno, da ima programska oprema visoko stopnjo zanesljivosti,
- vmesne, delne verzije niso dovolj za uporabo in je za uporabo potreben celoten izdelek,

- ni možnosti sprotnega preverjanja funkcionalnosti s strani končnih uporabnikov.

Gladkova navaja, da se slapovni pristop razvoja programske opreme uporabi kadar je pomembno, da ima programska oprema visoko stopnjo zanesljivosti. Menim, da naj visoka stopnja zanesljivosti ne bi bila kriterij za odločanje med slapovnim in agilnim pristopom. Saj naročnik na koncu potrebuje vedno visoko stopnjo zanesljivosti. Do zdaj mi še nihče ni specificiral zahteve, ki je pogojno uporabna.

2.3 Razlike med slapovnim in agilnim pristopom

V tabeli 1 sem povzela bistvene značilnosti in razlike med obema pristopoma, ki smo jih uporabili v analiziranem projektu.

Tabela 1: Primerjava slapovnega in agilnega pristopa

Področje	**Slapovni projektni pristop**	**Agilni pristop**
Opredelitev zahtev (specifikacije)	Vse zahteve so natančno specificirane v fazi snovanja. Pripravi jih naročnik, projektni manager ima svetovalno vlogo.	Narejen je seznam zahtev, ki pa še niso natančno specificirane. Natančno so specificirane le zahteve, ki bodo narejene v tekočem ciklu. Določi in pripravi jih skrbnik izdelka.
Časovno planiranje in kadrovanje	Se izvaja v fazi priprave. Projektni manager opredeli kadre in opremo. Linijski manager zagotovi ustrezne izvajalce, potrdi tehnologijo in čas izvedbe svojih ljudi. Izvajalci so različno obremenjeni, lahko sodelujejo pri celotnem projektu, lahko tudi le pri enem ali več delih projekta.	Priporočen je stalni tim, ki si samostojno razdeli naloge vsak cikel. Dolžina enega cikla je od 2-4 tedne. Vsi člani tima so polno zaposleni skozi celotno trajanje projekta Število in vsebina sestankov je za vsak cikel določena s strani metodologije..
Načrtovanje programske opreme	V fazi planiranja je celoten produkt že specificiran in načrt produkta narejen in potrjen.	Razvojna skupina se sproti dogovarja o načrtu cikla. Pomembno je le, da je ob koncu cikla na voljo uporaben izdelek.
Udeleženci in njihove vloge	Projektni manager je odgovoren za učinkovito izvajanje projekta. Planira, vodi tim in kontrolira izvedbo. Linijski manager (lahko direktor projektne pisarne) skrbi za strateško in poslovno umestitev projekta v podjetje. Skrbnik, ki izbere projektnega managerja, skrbi za potrjevanje plana , rešuje nesoglasja, potrjuje morebitne spremembe projekta. Tim, ki izvaja zahteve, ki mu jih dodeli projektni manager. Naročnik na koncu sprejme in potrdi izdelek.	Skrbnik metodologije je odgovoren za nemoten proces dela razvojnega tima, le-tega mora zaščititi pred vsemi motečimi dejavniki iz okolice. Skrbnik izdelka je odgovoren za specifikacijo in prednostno določitev zahtev za naslednji cikel. Razvojna skupina se med sabo dogovori, katero zahtevo bo kdo delal. To se dogovarja na 15 minutnih dnevnih sestankih.
Testiranje	Testiranje celotnega izdelka se izvaja v fazi testiranja, ki sledi fazi izvedbe, ko je izdelek že končan. Fazo testiranja izvaja tako razvojni tim, kot tudi naročnik, oz. končni uporabnik.	Testiranje se za vsako izdelano zahtevo izvaja sproti v okviru cikla. Testiranje izvaja razvojni tim. Ob zaključku cikla pa še skrbnik izdelka oz. uporabniki.

se nadaljuje

nadaljevanje

Področje	Slapovni projektni pristop	Agilni pristop
Management tveganj	V fazi planiranja projektni tim in strokovnjaki identificirajo možna tveganja projekta in jih analizirajo in ovrednotijo. V naslednjih fazah projekta pa jih kontrolirajo in ustrezno ukrepajo.	Na prvem sestanku za izdelavo plana cikla udeleženci pripravijo tabelo tveganj in jo ovrednotijo. Na vsakem naslednjem sestanku za izdelavo plana cikla tveganja prevrednotijo in vseskozi opazujejo zmanjševanje tveganj. Stroka si glede tega še ni enotna, saj eni trdijo, da management tveganja ni potreben, drugi pa ga še vedno uvrščajo med naloge agilnega tima (Veethil, 2013).
Obravnavanje sprememb	Ključne oz. nujne spremembe s vključijo tudi že med izvedbo projekta. Ostale spremembe se običajno vključijo na koncu projekta, ko so vse specificirane zahteve narejene in sprejete.	Sprememb običajno ni, saj so cikli natančno definirani. Za naloge v naslednjem ciklu se dogovorijo na začetku novega cikla, ki vsebuje prednostne naloge (ki so lahko tudi »spremembe«, če je le-ta prednostna naloga).
Implementacija (migracija) in začetek uporabe	Ko je faza testiranja in popravkov uspešno končana, se lahko začne faza implementacije. Sodeluje razvojni tim in IT oddelek naročnika.	Implementacija izdelka enega cikla je možna takoj po koncu cikla, saj mora biti rezultat cikla vedno uporaben. Sodeluje skrbnik izdelka in IT oddelek.
Reševanje nesoglasij	Projektni manager v fazi planiranja določi odgovornosti Nesoglasja najprej rešuje projektni manager, na kasnejši stopnji linijski manager.	Za reševanje nesoglasij je na prvi stopnji odgovoren razvojni tim, na slednji stopnji pa skrbnik metodologije, ki je tudi odgovoren za zaščito razvojnega tima pred motečimi dejavniki.

Faze snovanja agilni pristop ne vključuje, ker se osredotoča na razvoj in predvideva, da je odločitev o izvedbi projekta že sprejeta. Zato se podrobnejša opredelitev zahtev (specifikacij), ki jih tradicionalni projektni pristop uvršča v snovanje, pri agilnem pristopu izvede v sklopu planiranja oz. izvedbe projekta (v sklop posameznih ciklov). Podrobna opredelitev funkcij in ocena pričakovanih koristi teh funkcij za uporabnika se ponavlja v vsakem ciklu. Iz skupnega seznama zahtev se določi seznam nalog za cikel. Podrobno se definira le zahteve za prihajajoč cikel, kamor se lahko vključi tudi dodatne zahteve. Pogosto se zgodi, da določene zahteve nikoli niso narejene, ker niso prednostne in se ne izgublja časa z nepomembnimi nalogami. Pri slapovnem pristopu so vse zahteve že na začetku specificirane in razvijalci morajo upoštevati vsako podrobnost zahteve. Ključne spremembe se lahko vključijo tudi med projektom, a pomembno je zavedanje, da vsaka sprememba predstavlja dodaten strošek.

Pri agilnem pristopu naročnik oz. predstavnik naročnika kot skrbnik izdelka tesno sodelujeta z razvojnim timom med celotnim razvoja. Težavam se lažje izognejo in popravki dokumentacije so lahko narejeni v kratkem času, za razliko od slapovnega pristopa kjer odgovornost nosi projektni tim. Naročnik v fazi izvedbe skorajda ni aktiven.

Pri agilnem pristopu je razvojni proces ponavljajoč, saj se projekt izvaja v krajših ciklih katerih rezultat je uporaben izdelek. Vsaka nova različica izdelka je posodobljena z dodatno zahtevanimi funkcionalnostmi iz seznama zahtev. Za planiranje se porabi malo časa. Pri slapovnem pristopu se razvojni proces izvaja postopoma. Na koncu faze izvedbe se šele izdela uporaben izdelek.

Pri agilnem pristopu je faza testiranja vključena v cikel in to omogoča izdelavo delujočega izdelka. Pri slapovnem pristopu se faza testiranja izvede šele po fazi razvoja, ker posamezni deli nimajo delne funkcionalnosti, ki bi se jo dalo testirati. Kontroliranje razvoja se izvede lahko šele po zaključeni fazi izvedbe.

Pri agilnem pristopu se najprej razvijejo najpomembnejše funkcionalnosti. Ko se cikel konča naročnik lahko že testira zaključen del in nove funkcionalnosti so na voljo tako po namestitvi v okolje pri naročniku. Pri slapovnem pristopu so vse funkcionalnosti na razpolago naenkrat in sicer po zaključeni fazi izvedbe. Naročnik težje dodaja spremembe, ker le-to zahteva ponovno preučitev vseh faz.

Pri agilnem pristopu dokumentacija projekta ni tako formalizirana, je preprosta in razumljiva, cilj je delujoča programska oprema. Pri slapovnem pristopu je projekt odlično dokumentiran z različnimi vrstami dokumentov, vključno s uporabniškim priročnikom.

Pri agilnem pristopu so lahko manjši projekti izdelani in nameščeni zelo hitro. Težje je oceniti čas razvoja za velike projekte. Pri slapovnem pristopu je projekt natančno planiran in datum namestitve je točno določen, a pogosto prestavljen. (Gladkova, 2012).

Po mnenju Choudhurya (2012) so razlike med agilnim in slapovnim pristopom tudi:

- Glavna prednost agilne metodologije je možnost vgrajevanja z novih zahtevkov takoj po zaključenem ciklu. Pri slapovnem pristopu je težje spremeniti odločitve in razvoj, na katerih se dela v fazi izvedbe. Za vsako spremembo pri slapovnem pristopu je potreben popravek plana projekta in tudi vse potrebne odobritve. Popraviti je potrebno vso dokumentacijo.
- Pri agilnem pristopu se testira v kateremkoli delu razvojne faze. Tako so napake prej odkrite in spotoma odstranjene. Pri slapovnem pristopu, kjer se lahko testira le na koncu razvojne faze, je potem testiranje časovni zalogaj naenkrat.
- Agilni pristop dovoljuje prilagodljivost v primeru da naročnik doda spremembe pri zahtevkih in to je v veliki meri povezano z zadovoljstvom le-tega. Sprememba zahtevka v primeru slapovnega pristopa pa je povezana z nazadovanjem, ker slapovni pristop ne dovoljuje sprememb po končanju delovnega paketa.
- Pri agilnem pristopu je razvoj lahko bolj učinkovit v primerjavi s slapovnim pristopom. Čeprav oba pristopa omogočata možnost razčlenitve razvoja, pri slednjem manjkajo spremembe v zahtevkih v fazi izvajanja. Ker so pravila pri slapovnem pristopu določena pred začetkom projekta, je to ovira za nadaljnjo logično razčlenitev

razvoja. Če želimo, da je projekt bolj razčlenjen, je agilni pristop bolj naklonjen razvijalcem.

Menim, da je zmotna trditev Choudhurya v tretji točki, da slapovni pristop ne dovoljuje sprememb po končanju delovnega paketa, saj sem v praksi pogosto sodelovala pri vključitvi sprememb, katere je želel naročnik še dodati v izdelek. Pogosto se je to zgodilo, ko smo kot projektni tim prišli do točke, kjer specifikacija izdelka ni bila dovolj natančna za programiranje in smo potrebovali odgovor oz. obrazložitev naročnika. Takrat se je pokazalo, da je potrebna sprememba – razširitev specifikacije ali pa celo izbris zahtevka.

Tradicionalni pristop navadno definira in planira naloge ali spremembe, ki nikoli ne bodo razvite oziroma izdelane. Za razliko od agilnega pristopa, kjer je planirano le tisto, kar bo izpeljano in dostavljeno v finalni verziji projekta (Wysocki, 2006, stran 273).

Skrbnik metodologije pri agilnem projektu zagotavlja, da je proces razvoja skladen, zmanjšuje tveganja in pazi da so stroški in časovni okviri v okviru planiranega. Skrbniku metodologije je tako kot pri tradicionalni metodologiji pomembno, da naročniku izroči izdelek, ki vsebuje največjo možno dodano vrednost za končnega uporabnika.

Planiranje v agilnem pristopu predstavlja dejavnost, ki jo razvojni tim opravlja, ko jo potrebuje in sicer skozi ves čas trajanja projekta. Vsaka dodatna sprememba že planiranega pri slapovnem pristopu zahteva dodaten čas za analizo, dodatno planiranje in izvedbo. Potrebno pa je upoštevati, da lahko dodatna sprememba doda višjo končno dodano vrednost izdelku na eni strani in večji zaslužek projektnemu timu. Najpomembnejši kriterij za skrbnika projekta in skrbnika izdelka je poslovna vrednost končnega izdelka v primerjavi s planiranim in planiranimi upravičenimi spremembami.

3 ANALIZA PROJEKTA »CMS REFACTORING TURKEY«

3.1 Podjetje

Delala sem v mednarodnem podjetju, ki ima podjetja v 40 državah po vsem svetu. Ukvarja se s finančnimi storitvami in s tem podpira prodajo izdelkov matičnega podjetja, ki proizvaja avtomobile, avtobuse in tovornjake. Podjetje v Sloveniji je imelo samostojni oddelek za razvoj programske opreme za vodenje leasing pogodb. Oddelek je bil odgovoren matičnemu podjetju v Nemčiji. Od leta 2001 smo samostojno razvijali kompleksno aplikacijo za vodenje leasing pogodb za 4 hčerinska podjetja in sicer v Sloveniji, Grčiji, Turčiji in na Hrvaškem.

3.2 Ozadje, namen in cilji obravnavanega projekta

Naš oddelek ja razvijal programsko opremo za štiri različne evropske trge v okviru koncerna. Razvojno okolje programske opreme je Microsoft Visual Fox Pro, ki je primerno za zgradbo baz podatkov vseh velikosti. Zadnja različica je 9.0 in je bila izdana leta 2007.

Končni datum podaljšane splošne podpore pa je bil 13.01.2015. Zaradi tega smo v oddelku že leta 2003 začeli pogovarjati in razvijati strategije nadgradnje razvojnega okolja.

To je bil razlog zakaj je bilo potrebno pridobiti interes in finančna sredstva s strani vseh 4 podjetij. Leta 2006 so se pri strategiji programske opreme odločili, da preidejo na novo platformo in prenesejo kasneje vse funkcionalnosti nanjo. Ta projekt se je po pol leta neuspešno končal. V naslednjih letih so se v našem oddelku menjavali vodilni in nihče ni imel vizije, niti interesa da bi nadgradili programsko opremo.

Tako smo leta 2011 dobili informacijo, da bo podjetje nehalo ponujati finančne storitve na slovenskem trgu. To je za nas pomenil zaključek novega razvoja za slovensko podjetje. Tudi grško podjetje ni imelo interesa za nadgradnjo, saj so vedeli, da se bo do konca leta 2014 iztekla večina pogodb.

Največji interes so pokazali iz turškega podjetja, saj so imeli ogromen porast portfelja in pogodb. Turško podjetje je v tem obdobju prikazovalo ekstremno visoko rast tržnega deleža in zaradi tega so potrebovali čim bolj stabilen, hiter in zanesljiv programski paket. To je bil razlog, da je bila naša turška stranka največji interesent za nadgradnjo programskega paketa, saj se je ob povečanem številu uporabnikov, velikem številu pogodb in velikem številu novih podatkov, soočala z omejitvami okolja Microsoft Visual Fox Pro. Modeliranje podatkov je bilo problematično, saj je bilo v eni tabeli tudi do 256 podatkov. Podatkovni model ni deloval normalno, podatkov je bilo preveč za pravilno delovanje, pojavljali so se primeri podvajanja. To je bila pot v propad podatkovne zbirke. Večslojna arhitektura, ki jo lahko na razno razne načine apliciraš je bila rešitev. CSLA.NET je aplikacijsko ogrodje, katerega temeljni doprinos je organizacija poslovne logike v skladu z ločevanjem modulov. Hkrati pa ima nastavke za povezavo s tehničnimi plastmi.

Naš cilj je bil sprogramirati novo tehnično rešitev, ki bi zagotavljala enostavno vzdrževanje in enostavne nadgradnje, kar star program ni omogočal. Prav tako je pomembno ločiti poslovno logiko od uporabniškega vmesnika. Temeljno je, da se da poslovno logiko avtomatsko testirat. S tem se pridobi na udobju in primernosti vzdrževanja in nadaljnjih nadgradenj programske opreme.

V začetku leta 2012 smo se lotili dela, ki ga opisujem in analiziram v mojem diplomskem delu. Imeli smo kar nekaj izkušenj z agilnim pristopom pri manjših projektih v preteklosti. Vedno so se končali uspešno in z zaključenim, uporabnim programskim paketom. Z izkušnjami iz preteklih let smo bili sigurni v uspešen zaključek projekta.

3.3 Potek projekta »CMS Refactoring Turkey«

Primarni razlog za izvedbo projekta je bil tehnične narave in ne funkcionalne narave. V celoti smo ga imenovali preoblikovanje kode in poslovne logike.

Februarja 2012 smo predstavili projekt s predlagano metodologijo našemu vodji, saj smo se vsi zavedali, da trenutnemu razvojnemu okolju za našo programsko opremo splošna podpora poteče januarja 2015. Potrebno je bilo najti najboljšo rešitev. Predlagana rešitev je

bila sprejeta le s strani turške stranke, ki se je najbolj soočala s tehničnimi težavami pri uporabi programske opreme, saj se je turški trg najhitreje širil. Samo oni so v trenutni situaciji na trgu lahko investirali v tehnični razvoj oz. nadgradnjo.

Vodja projekta je turški stranki predstavil rešitev le kot izboljšave funkcionalnosti, s katerimi so imeli težave. Na začetku nismo bili vključeni v razgovore in tako kot razvojni tim sploh nismo vedeli, kaj točno je bilo v fazi ideje dogovorjeno. Naš vodja nam je le sporočil, da bomo projekt začeli.

V začetku projekta je tehnični vodja oddelka predlagal, da začnemo projekt po agilni metodi. Agilna metoda omogoča postopno preoblikovanje kode po modulih s čimer bi zagotovili takojšnjo uporabo izboljšanih funkcionalnosti. Ker je bilo potrebno prepisati celotno funkcionalnost programske opreme, smo upravi predlagali, da začnemo z moduli, ki podpirajo poslovne funkcionalnosti, kjer imajo največ težav pri vsakodnevnem delu. Cilj je bil, da po delih preidemo na novejše razvojno okolje in sicer Microsoft .NET Framework, ki je ogrodje za razvijanje programske opreme na Microsoft operacijskih sistemih.

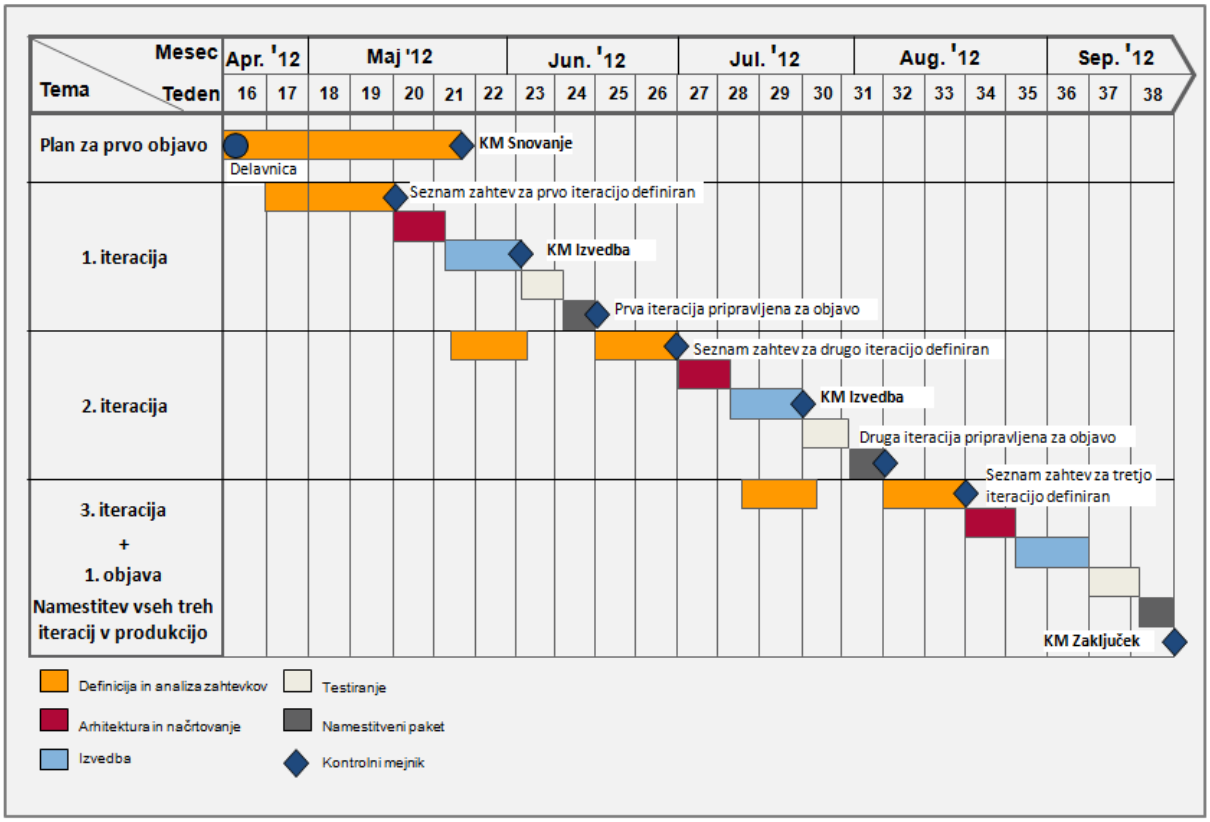
Odločili smo se za uporabo objektno orientiranega načina razvoja programske opreme. Objektno orientiran način pomeni, da programska oprema deluje kot celota, a je sestavljena iz številnih samostojnih modulov oz. objektov. Eden temeljnih principov za objektno orientiran pristop za izgradnjo aplikacij je ločitev modulov (angl. *Separation of concern*). Hkrati pa je to eden od dveh temeljnih načinov kako kompleksno aplikacijo narediš obvladljivo. In naša aplikacija je z leti postala zelo kompleksna, saj je podpora financiranju. Ločitev modulov pomeni ločevanje računalniškega programa na module posameznih tem oz. funkcionalnosti. Večslojno programiranje npr. za predstavitev – uporabniški del, poslovno logiko, dostop do podatkov naredi računalniški program pregleden, enostaven za razvoj in vzdrževanje. Če so moduli dobro razdeljeni, so posamezni deli lahko ponovno uporabljeni in tudi nadgrajeni neodvisno. Zaradi takšnega načina programiranja, se je uporaba agilnega pristopa za nadgradnjo obstoječe programske opreme zdela še primernejša.

V preteklosti smo tako prepisali in modernizirali že dva modula v Sloveniji in na Hrvaškem in sicer modul za reprogram leasing pogodb in kalkulacijo. Na enak način smo planirali tudi modernizacijo aplikacije v Turčiji.

Slika 7 prikazuje plan projekta glede na agilni pristop in iteracije. Ob hitrem pogledu zgleda slika kot slapovni pristop, a če se osredotočimo na barve, vidimo, kako se iteracije in njih vsebina ponavljajo in zaključujejo z delujočo objavo ob vsakem mejniku.

Potek prve iteracije se je določal s strani managementa obeh podjetij. Uporabniki v Turčiji so naredili seznam, kjer bi bilo potrebno najhitreje nadgraditi programsko opremo, saj so imeli vsakodnevne težave pri delu. Na osnovi tega smo določili obseg projekta.

Slika 7: Plan prve objave



Vir: Interna projektna dokumentacija, 2012.

Ocenjena vrednost projekta je bila 850 tisoč evrov. Tako smo padli v kategorijo »B« interne Houston kategorizacije projektov in bili s tem zavezani k upoštevanju internih Houston standardov. Za odobritev projekta, je bilo potrebno pridobiti odobritev regionalne evropske projektne uprave (angl. *European Project Board*) (v nadaljevanju EPB). Da pa bi EPB odobril projekt, je bilo potrebno pripraviti vso potrebno in zahtevano projektno dokumentacijo.

Ker programske opreme ni bilo možno nadgraditi brez predhodne tehnične nadgradnje je razvojna skupina pripravila plan za tehnično nadgradnjo oz. plan za razvoj tehnične platforme v prvi iteraciji, z vključenimi nekaj funkcionalnostmi, ki so bile nujno potrebne. Tak plan je bil sprejet in bil tudi odobren s strani EPB (Evropski projektni odbor) v Nemčiji.

To je faza snovanja. Pri slapovnem pristopu zahteve pripravi naročnik, pri agilnem pa zahteve pripravi skrbnik izdelka, ki je tudi s strani naročnika. V našem primeru smo zahteve za prvo in drugo iteracijo pripravili v razvojnem timu, zaradi specifičnosti tehničnih zahtev za pripravo platforme za razvoj programske opreme. Za natančno sledenje enemu ali drugemu pristopu, bi priprava platforme morala biti gotova že pred začetkom projekta. Ostale zahteve je pripravil skrbnik izdelka v sodelovanju s članom razvojnega

tima, kar je tudi kombinacija agilnega in slapovnega pristopa, kjer projektni manager svetuje naročniku.

3.4 Način izvedbe

3.4.1 Projektna dokumentacija

Potrebno je bilo pridobiti potrditve za agilni pristop razvoja programske opreme. Pravila po HOUSTON metodologiji niso predvidevala agilnega pristopa, zato smo morali v veliki meri improvizirat in se dodatno ukvarjat s birokratskimi zadevami in pridobivanjem odobritev, da krenemo naprej, brez določenih kakovostnih mejnikov kot opisano na sliki 8 in poglavju 3.5.1. To je časovno zelo obremenjevalo naš majhen tim.

V koncernu so na hitro dodali novo metodologijo brez konkretnih upoštevanj agilne metodologije na primer v točkah kontrolnih mejnikov. Se pravi ob kontrolnih mejnikih faze snovanja so bili uradno še vedno potrebni enaki dokumenti kot pri HOUSTON slapovnem pristopu. Zaradi tega smo bili dodatno obremenjeni s pripravo dokumentacije. Prav tako so od nas zahtevali dokumentacijo, ki pri agilnem pristopu na začetku sploh ni planirana. Tako so bila moja naloga tudi pogajanja s kolegi v tujini, ki so bili odgovorni za preverjanje kontrolnih mejnikov, da so nam tudi brez določene dokumentacije dali zeleno luč.

Tak način priprave projektne dokumentacije je bolj podoben slapovnem, kot agilnem pristopu, saj agilni ne predvideva kontrolnih mejnikov z dokumentacijo v okviru ene iteracije, ampak le mejnik na koncu iteracije in sicer v obliki delujočega izdelka.

3.4.2 Področja znanja slapovnega pristopa pri kombiniranju z agilnim

Na kick – off delavnici dva meseca kasneje, so bila predstavljena področja znanja iz PMBOK. Za vsako področje znanja je bila opisana vsebina, prikazana na sliki 8. Pregled področij znanj je tipično za slapovni pristop, ne za agilni. Tu smo v področja znanja slapovnega pristopa opisovali agilni pristop. Na tej sliki je razvidno kombiniranje obeh pristopov.

Tudi kick – off delavnica ni opisana pri agilnem pristopu, ampak pri slapovnem. Agilni projekt se začne s sestankom za izdelavo plana iteracije (angl. *Sprint Planning Meeting*)

Slika 8: Pregled področij znanj na kick-off delavnici

Vir: Interna projektna dokumentacija, 2012.

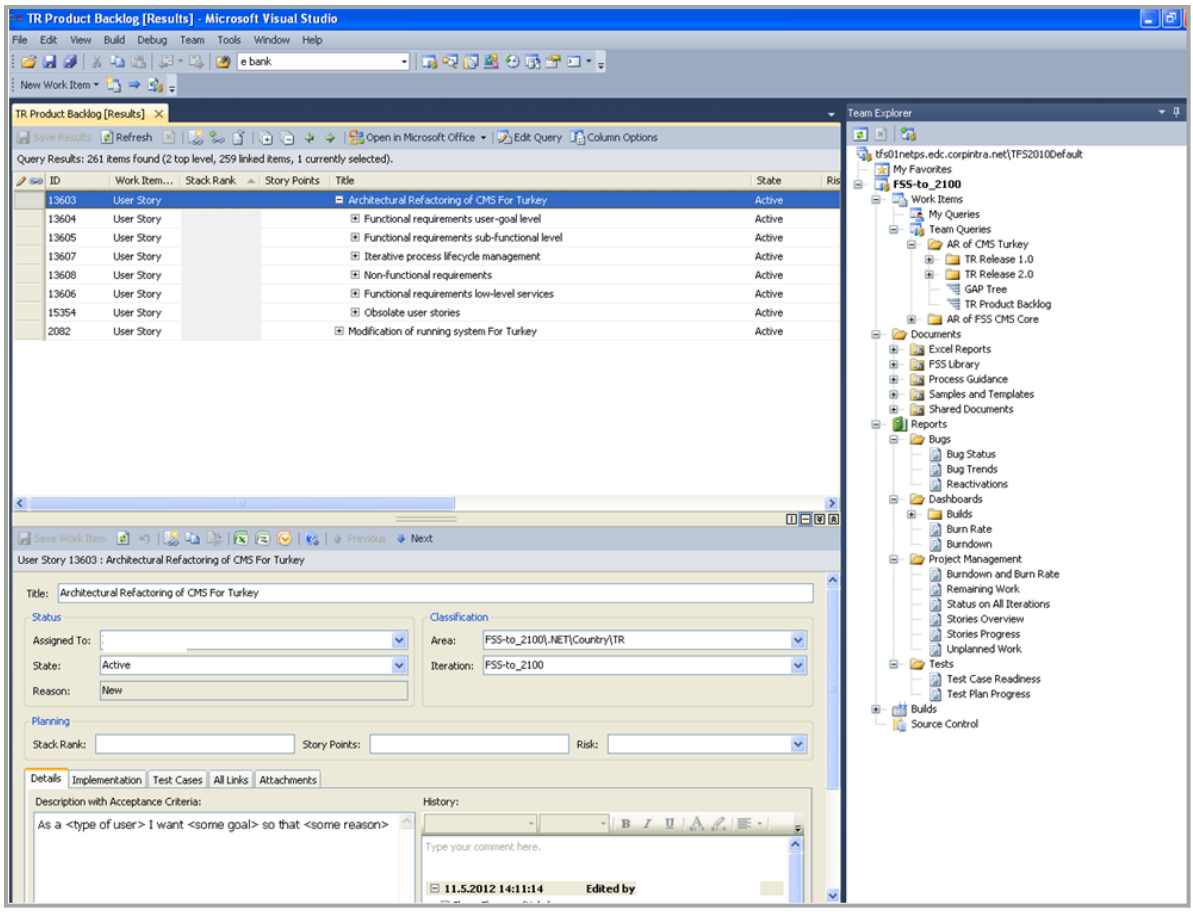
3.4.3 Seznam zahtev

V skladu s SCRUM agilno metodologijo smo zahtevke zbirali v aplikaciji TFS Team Foundation Server. To so bili funkcionalni zahtevki za uporabnika, kot tudi tehnični zahtevki za pripravo platforme. Sliki 9 in 10 prikazujeta seznam vseh zahtev oz. uporabniških zgodb, ki so že sortirane glede na funkcionalnosti. Vključen je podatek o spremembah, kdaj in kdo je spremembo shranil in čas za razvoj. Eni uporabniški zgodbi se lahko priloži dodaten dokument kot prilogo. Lahko se preveri podatek o stanju uporabniške zgodbe in primeri uporabniškega testiranja.

Velikost oz. zahtevnost uporabniških zgodb smo ocenjevali s številom točk Fibonaccijevega zaporedja in sicer 0, 1, 3, 5, 8, 13, 21, 34,…s tem, da je 1 točka pomenila 8 ur. To je popolnoma v skladu s Scrum pristopom.

Bazo zahtevkov je osveževal razvijalec in ne, kot je predpisano po Scrum pristopu, skrbnik izdelka. To pa zato, ker skrbnik izdelka ni imel dovolj znanj o funkcionalnostih, niti o tehnoloških zahtevah.

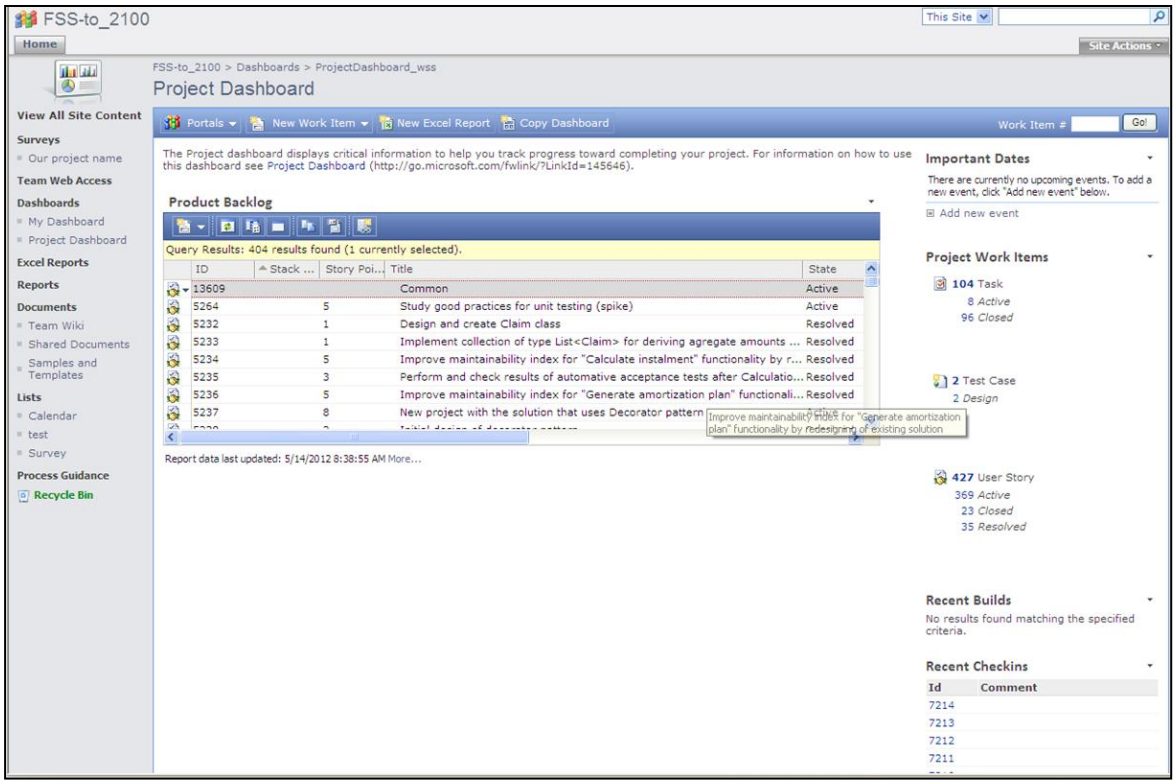
Slika 9: Zajem zaslona seznama zahtev



Vir: Daimler AG, Product Backlog, 2012.

Uporabniške zgodbe smo zbirali glede na funkcionalnost in jih tematsko združevali v skupine (angl. *epic).* Uporabniške zgodbe smo pripravljali glede na obstoječe funkcionalnosti, ki smo jih poznali in smo vedeli, da jih stranka uporablja pri vsakdanjem delu. Ker smo se na ta projekt pripravljali že prej, smo določene ocene imeli na razpolago že predno smo določili prvo iteracijo. Ta korak ni bil v skladu niti s slapovnim, niti agilnim pristopom, saj nikjer razvojni tim ne določa oz. piše uporabniških zgodb.

Slika 10: Zajem zaslona seznama zahtev kot so ga lahko videli vsi udeleženi v projektu



Vir: Daimler AG, Product Backlog, 2012.

3.4.4 Tedenski pregled statusa projekta

Imeli smo tedenski pregled statusa projekta oz. pregled področij znanj, ki je obvezen del projekta po slapovnem pristopu. Priprava tedenskega pregleda je naloga projektne pisarne, ki pa pravzaprav ni bila določena, tako je razvojni tim pripravljal tudi tedenski pregled statusa projekta. Poleg tega smo imeli tudi kratke dnevne sestanke, vendar le v okviru razvojnega tima. Tu nismo sledili niti slapovnem, niti agilnem pristopu. Saj bi na dnevnem sestanku morala biti prisotna tudi skrbnik metodologije in skrbnik izdelka, a sta bila prisotna le na tedenskem sestanku, ki pa ga agilni pristop sploh ne predvideva.

3.5 Uporabljena metoda projektnega managementa

Houston metodo projektnega managementa bom opisala, ker je pravilnik mednarodnega podjetja iz katerega prihaja izbrani analizirani projekt določal sledenje tej metodi in ker bom v nadaljnji analizi opisovala, kje smo pravilno in tudi uporabno sledili tej metodologiji in kje smo naredili napake.

Houston metoda projektnega managementa je bila razvita v IT oddelku avtomobilske multinacionalke na osnovi priporočil Project Management Instituta. V koncernu je to osnova za management vodenje projektov. Vsi projekti, nad določeno vrednostjo proračuna, morajo slediti tej metodologiji. Pravila Houston Project Guide (Daimler AG,

2011a) ponujajo strukturo v obliki skupin procesov in znanj, z uporabo procesnih dokumentov, obrazcev in obrazcev s primeri vsebine, kontrolnih seznamov in delovnih pripomočkov. Vse to projektni manager, ki je zaposlen v multinacionalki, lahko poišče na intranetu.

Faze projekta, kot tudi področja znanja so v popolnosti skladne s priporočili PMBOK in sem jih prikazala v prvem poglavju in točki 1.3.

Pri Houston metodologiji so označili področja znanja z barvami in ikonami (Slika 11) z namenom, da se že na prvi pogled vidi v katerem področju smo aktivni in katera pravila, dokumentacijo in vsebino obravnavamo. Ikone se pojavljajo v celotnem priročniku in tudi v vseh obveznih pripadajočih dokumentih, ki jih predpisuje obravnavano mednarodno podjetje pri projektnem managementu. Primer uporabe je prikazan na sliki 8 v poglavju Tedenski pregled statusa projekta.

Slika 11: Barvni simboli za področja znanja pri Houston metodologiji

Vir: Daimler AG, Houston Project Guide,2011a.

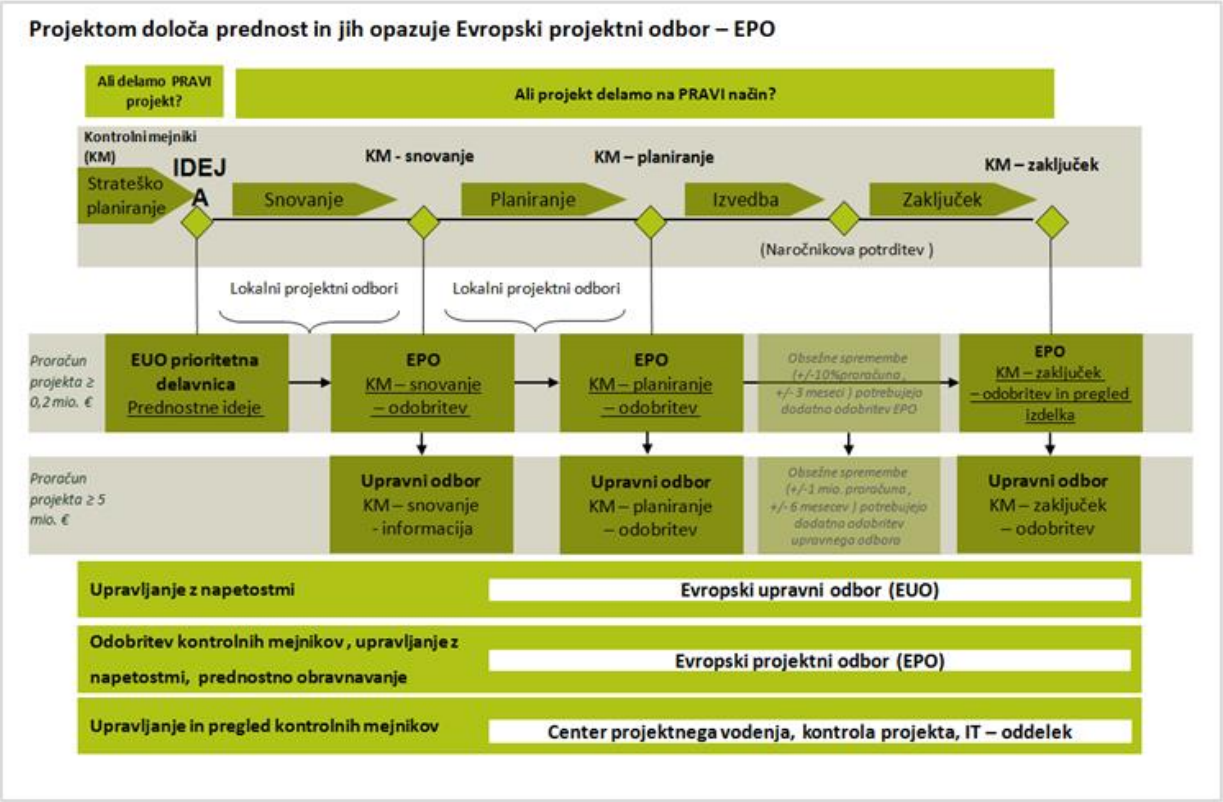
Houston Project Guide multinacionalke je namenjen projektnim managerjem, projektni pisarni in vsem, ki imajo kakršnokoli podporno in svetovalno vlogo v projektu. Multinacionalka je v okviru orodja za projektni management na intranetu poskrbela tudi za vse potrebne osnutke dokumentov, katerih uporaba je obvezna pri projektnem vodenju. Projektni management v podjetju igra pomembno vlogo in zato je tudi dobro podprt s primeri uporabe in obrazložitvami.

3.5.1 Podrobnejše opredeljen proces izvedbe projekta z odločevalnimi mejniki po interni metodologiji

Slika 12 prikazuje proces izvedbe projekta s kontrolnimi mejniki (v nadaljevanju KM) in hkrati odločevalnimi mejniki (angl. *Quality Gates*). V podjetju je med vsako fazo projekta kontrolni oz. odločevalni mejnik, ki potrebuje odobritev določenega odbora v podjetju, da se ena faza zaključi in preide v naslednjo fazo projekta. Odbor za odobritev prehoda v

naslednjo fazo projekta je za različne velikosti projektov različen. Da projekt lahko preide v fazo snovanja, je najprej potrebna predstavitev ideje na prioritetni delavnici evropskega upravnega odbora (v nadaljevanju EUO), kjer se planira strategija v podjetju in določa strateško pomembne projekte. Vse predstavitve in obrazce, ki so potrebni ob kontrolnih mejnikih za pregled v centru projektnega vodenja pripravi projektna pisarna (v nadaljevanju PMO) projekta. Tako pred vsakim prehodom v novo fazo projekt potrebuje odobritev Evropskega projektnega odbora (v nadaljevanju EPO) oz. EUO odvisno od planiranega proračuna.

Slika 12: Evropski proces izvedbe projekta z odločevalnimi mejniki

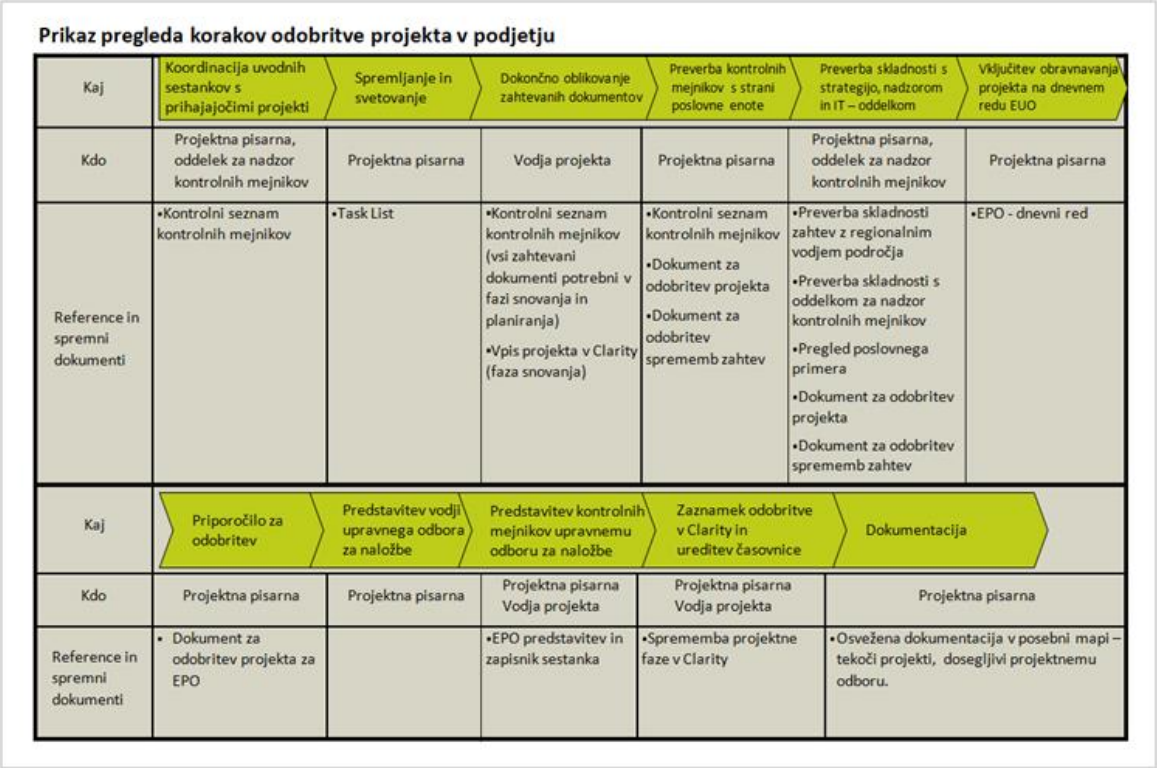


Vir: Daimler AG, Governance Guidline, 2011b.

3.5.2 Proces odobritve projekta po interni metodologij – koraki

Vsak kontrolni mejnik vsebuje pregled in kontrolo dokumentov potrebnih za prejšnjo fazo, kar prikazuje slika 13. Za potrditev kontrolnega mejnika so odgovorni posebni oddelki, ki se ukvarjajo le s pregledovanjem kontrolnih mejnikov in dokumentov za le-te. Kontrolni mejniki zahtevajo sklop dokumentov, ki vsebujejo tudi posebne zahtevke s spiski preverjanj (angl. *check list*). Spiski preverjanj so sestavljeni s strani strokovnjakov in organov odločanja. Glede na njihov pregled in odločitev na podlagi le-tega, se projekt lahko prekine, začasno ustavi, ali odobri in s tem omogoči normalno nadaljevanje.

Slika 13: Proces odobritve projekta – koraki



Prikaz pregleda korakov odobritve projekta v podjetju

Kaj	Koordinacija uvodnih sestankov s prihajajočimi projekti	Spremljanje in svetovanje	Dokončno oblikovanje zahtevanih dokumentov	Preverba kontrolnih mejnikov s strani poslovne enote	Preverba skladnosti s strategijo, nadzorom in IT – oddelkom	Vključitev obravnavanja projekta na dnevnem redu EUO
Kdo	Projektna pisarna, oddelek za nadzor kontrolnih mejnikov	Projektna pisarna	Vodja projekta	Projektna pisarna	Projektna pisarna, oddelek za nadzor kontrolnih mejnikov	Projektna pisarna
Reference in spremni dokumenti	•Kontrolni seznam kontrolnih mejnikov	•Task List	•Kontrolni seznam kontrolnih mejnikov (vsi zahtevani dokumenti potrebni v fazi snovanja in planiranja) •Vpis projekta v Clarity (faza snovanja)	•Kontrolni seznam kontrolnih mejnikov •Dokument za odobritev projekta •Dokument za odobritev sprememb zahtev	•Preverba skladnosti zahtev z regionalnim vodjem področja •Preverba skladnosti s oddelkom za nadzor kontrolnih mejnikov •Pregled poslovnega primera •Dokument za odobritev projekta •Dokument za odobritev sprememb zahtev	•EPO - dnevni red
Kaj	Priporočilo za odobritev	Predstavitev vodji upravnega odbora za naložbe	Predstavitev kontrolnih mejnikov upravnemu odboru za naložbe	Zaznamek odobritve v Clarity in ureditev časovnice	Dokumentacija	
Kdo	Projektna pisarna	Projektna pisarna	Projektna pisarna Vodja projekta	Projektna pisarna Vodja projekta	Projektna pisarna	
Reference in spremni dokumenti	• Dokument za odobritev projekta za EPO		•EPO predstavitev in zapisnik sestanka	•Sprememba projektne faze v Clarity	•Osvežena dokumentacija v posebni mapi – tekoči projekti, dosegljivi projektnemu odboru.	

Vir: Daimler AG, Governance Guidline 2011b.

Podjetje na začetku klasificira projekt. In sicer glede na planiran proračun. Od tega je odvisen upravni odbor in projektni tim, kvalifikacija projektnega managerja, katera področja znanja se bodo uporabila, poročanje in kakšen bo odobritveni proces. Za kategorizacijo se uporablja "*Houston kategorizacijski plan*", ki vključuje natančno določena pravila ter ponderirane kriterije in tako avtomatsko določi kategorijo. Uporablja se tri kategorije in sicer:

- A – projekti
- B – projekti
- C – projekti

Projekt je v vsakem primeru A – projekt, če je proračun večji kot 5 mio. evrov, če so posebno kratki roki, ki so povezani z zakonskimi spremembami oz. vključuje visoka tveganja pravočasnega dokončanja. Projekt se običajno klasificira kot C – projekt v primeru proračuna manjšega od 0,5 mio evra z majhnim tveganjem, s timom manjšim od 5 ljudi.

Projekti s proračunom manjšim od 200 tisoč evrov se obravnavajo kot poslovni primer. Vsi projekti s planiranim proračunom večjim od 200 tisoč evrov morajo upoštevati Houston pravila določena v podjetju in slediti določenim pravilom.

3.6 Težave povezane z izbiro metodologije razvoja

3.6.1 Nepoznavanje oz. nepripravljenost v koncernu na agilni SCRUM pristop

Sobivanju agilnega in Houston slapovnega pristopa na enem projektu je sledilo dolgotrajno dogovarjanje in analiziranje, kako bi lahko speljali projekt na agilen način, ki je bil smiseln in možen na takšnem tehnično zahtevnem projektu.

V času izvajanja projekta smo ugotovili, da je prenos iz slapovnega pristopa v agilni pristop možen, obratno pa žal ne oz. je projekt potrebno začeti znova s fazo snovanja in priprave, kjer se pripravijo vse zahteve. Agilni pristop smo uporabili le notranje, navzven smo morali še vedno slediti Houston kontrolnim mejnikom, ki takrat še niso bili prilagojeni za agilni pristop.

3.6.2 Določitev obsega projekta

Pri agilnem pristopu se važnejšim zahtevam določi prednostno stopnjo in se jih natančno definira pred začetkom cikla. In to smo naredili. Obseg celotnega projekta je bil le okvirno podan. Pri slapovnem pristopu bi bilo potrebno pripraviti vso dokumentacijo za vse zahteve še pred fazo izvedbe. To ni bilo narejeno. Tako lahko rečem, da je bila za definiranje zahtev uporabljena agilna metoda. Moje mnenje je, da je bil v tej fazi uporabljen agilni pristop, ker je bilo bolj enostavno definirat le del zahtev. Po analizi primera, ne morem potrdit, da je to kombiniranje agilnega in slapovnega, ker je bil tu izbran le lažji in hitrejši postopek definiranja zahtev.

3.6.3 Neizkušenost skrbnika izdelka in izpolnitev njegovih nalog

Za skrbnika izdelka je bila izbrana oseba, ki ni imela dovolj izkušenj za to funkcijo. Agilnega pristopa ni poznala, še nikoli ni delovala v vlogi skrbnika izdelka, njene vsakodnevne obveznosti so bile preobsežne, da bi lahko tvorno sodelovala kot skrbnik izdelka. Težave so se dodatno pojavile zaradi neavtonomnosti pri odločanju in nezmožnosti prevzemanja nalog dejanskega skrbnika. Ves čas so ji nadrejeni spreminjali navodila, se niso strinjali z njenimi odločitvami in ji nalagali dodatne obveznosti in s tem je izgubila stik z nami, kot razvojnim timom. Skrbnik izdelka lahko vzdržuje in pripravlja funkcionalne uporabniške zgodbe. Ne more pa vzdrževat in pripravljat tehničnih uporabniških zgodb. Zato je definiranje tehničnega dela uporabniških zgodb prevzel tehnični vodja razvojnega tima. To ni v skladu z agilnim pristopom, saj mora biti le skrbnik izdelka odgovoren in zadolžen za definiranje zahtev oz. uporabniških zgodb.

Funkcionirali bi le tako, da bi imeli kvalificiranega skrbnika izdelka, ki bi skrbel za definiranje uporabniških zgodb najbolj prioritetnih stvari. Tako pa je skrbnik izdelka vsakodnevno dodajal nove funkcionalnosti, katere so bile z vidika težav, ki so jih imeli pri

delu z obstoječo programsko opremo nepomembne. Tak primer je bil na primer spletna aplikacija za vpogled plana plačil za stranke in podobne visoko tehnološke funkcionalnosti. V realnosti pa že kalkulacija za stranko in kreiranje plana plačil, ki so osnovna funkcionalnost, v podjetju ni delovalo, kot bi moralo.

Železni trikotnik omenjen v točki 1.3 prikazuje odvisnost kakovosti od obsega, cene in časa. Ta odvisnost se je pri našem projektu zanemarila, ko je naročnik želel spremembo ene omejitve, brez spremembe druge omejitve. Želel je povečati obseg ob nespremenjenem številu ljudi (cena) in nespremenjenem času. Se pravi, da bi z nespremenjenim številom programerjev in nespremenjenim številom delovnih dni, izdelali več funkcionalnosti v programski opremi. Na tak način kakovost v enaki meri ne bi bila več zagotovljena.

V današnjih projektnih okoljih, morajo projektni managerji razširiti svojo perspektivo tudi na druge kriterije, da zadovoljijo naročnika in da zagotovijo dobre poslovne rezultate. (Duggal, 2010).

3.6.4 Obseg projekta in cilj

Prvi cikel je bil planiran za daljše obdobje od enega meseca, kar ni v skladu z agilnim pristopom, ki predpisuje cikle dolge od 2 – 4 tedne. Ob tem je cikel vključeval prevelik zalogaj tehnične nadgradnje in premalo uporabnih funkcionalnosti. Tehnična nadgradnja bi morala biti pripravljena še predno smo začeli s projektom, se pravi pred planiranjem prve iteracije. Saj bi le tako lahko po prvem ciklu uporabnikom dejansko dostavili programsko opremo z novimi delujočimi in uporabnimi funkcionalnostmi. Ker je prvi cikel vključeval le tehnično nadgradnjo in nobene vidne funkcionalne izboljšave, so bili uporabniki razočarani, saj so pričakovali uporabno vrednost po zaključku cikla. Vsaj tako smo jim zagotavljali v skladu s Scrum pristopom, ki v teoriji na koncu iteracije predvideva izdelek z uporabno vrednostjo. V drugi iteraciji smo v razvojnem timu že pripravili prvi uporabniški vmesnik. Zataknilo se je pri videzu. Uporabniki niso bili zadovoljni z videzom aplikacije. Funkcionalnosti sploh niso preverjali.

Tako v prvem ciklu nismo sledili agilnem pristopu, saj je pravilo, da je potrebno na koncu cikla uporabniku predati delujoč izdelek. Objava rezultata drugega cikla pa ni bila sprejeta z zadovoljstvom, ker videz ni bil, kot so si predstavljali, kljub temu da je bil izdelek delujoč. V seznamu zahtev za cikel ni bilo uporabniške zgodbe za videz programske opreme, oz. uporabniškega zaslona. V prihodnje je potrebno upoštevati tudi prikaz uporabniškega zaslona, saj je očitno, da predstava uporabnika ni enaka predstavi programerja.

3.6.5 Dnevni sestanek za sprotni pregled dela in usklajevanje lokacij

Dnevnega scrum sestanka nismo imeli, ker niti skrbnik izdelka, niti skrbnik metodologije nista imela časa vsak dan 15 minut sodelovat. Zato smo imeli dnevni sestanek le v okviru razvojnega tima. Enkrat na teden pa klasični projektni sestanek za pregled statusa vseh področij znanj projekta, kjer so bili poleg razvojnega tima, prisotni skrbnik izdelka,

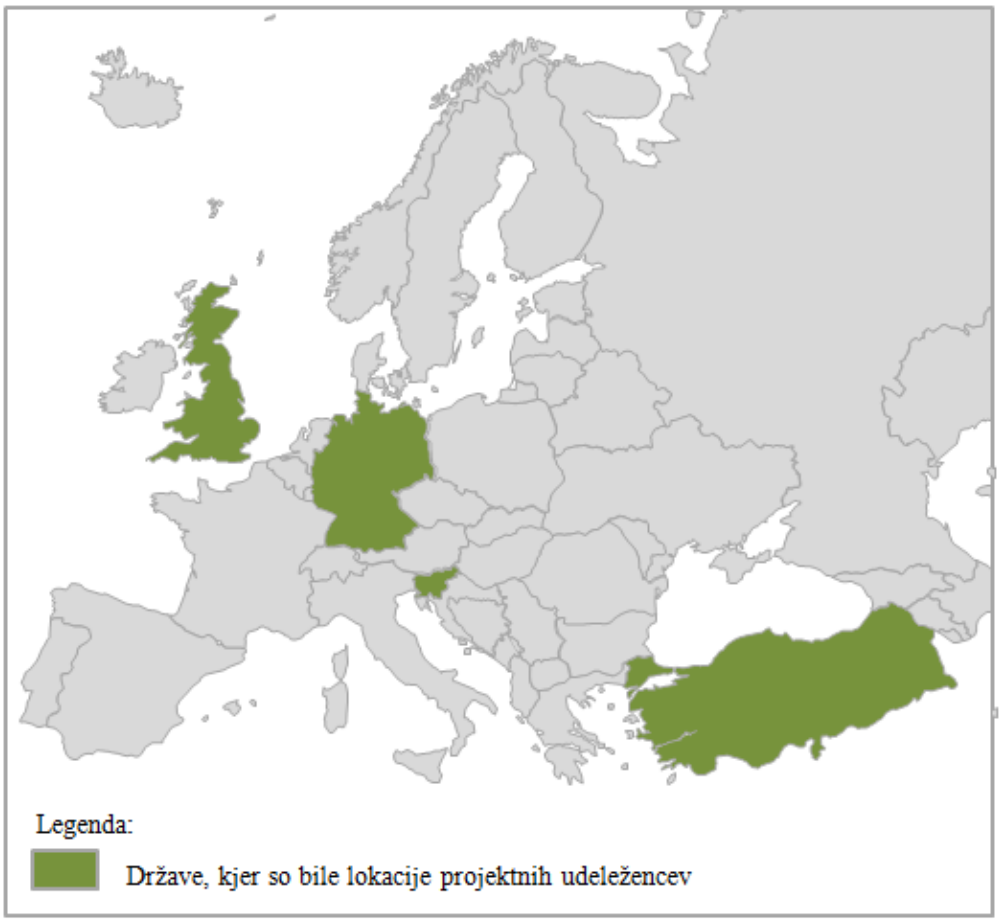
skrbnik metodologije, vodja projekta, in včasih vodja oddelka. To so bili le konferenčni klici, saj smo usklajevali štiri lokacije (Slika 14) v treh časovnih pasovih po Evropi.

Ob tej točki lahko potrdim ugotovitev, da je šlo za kombiniranje agilnega in slapovnega pristopa, saj smo vključili in priredili komunikacijo, kot ju predlagata oba pristopa.

3.6.6 Menjava skrbnika metodologije na začetku projekta in lokacija Scrum tima

Skrbnik metodologije je bil vodja našega tima v Stuttgartu, kar tudi ni ustrezalo natančnim navodilom metodologije Scrum. Imel je še vrsto drugih odgovornosti, ki niso bile del tega projekta, ampak druga dela v Nemčiji. Kasneje se je za skrbnika metodologije določil nov vodja projekta, ki še ni delal po agilni metodologiji, lociran je bil v Veliki Britaniji. Ob tem smo se dodatno soočili z lokacijskimi usklajevanji. Skrbnik metodologije je imel pisarno v Nemčiji oz. Veliki Britaniji, skrbnik izdelka v Turčiji, razvojna skupina v Sloveniji.

Slika 14: Lokacije udeležencev na projektu



Tranzicija vodenja je še dodatno otežila delo, saj je vsak potreboval kar nekaj časa, da je s pomočjo znanja in informacij razvojnega tima sploh dobil vpogled v velikost in pomembnost projekta. S tem smo izgubljali dragocen čas.

Skrbnik metodologije ni opravljal točno določenih nalog, ki jih po Scrumu ima in sicer da ves proces poteka nemoteno. Skrbnik metodologije v našem primeru ni branil razvojnega tima pred zunanjimi vplivi. Ukvarjat bi se moral s tem, da postane razvojni tim čim bolj produktiven, da dvigne kvaliteto izdelka in mora skrbeti, da se upoštevajo Scrum pravila. Njegova glavna naloga je odstranjevanje ovir in pomoč pri premagovanju problemov. Nalogo, ki jo je dejansko opravljal je bila klasična vloga projektnega vodje. Skrbnik metodologije bi moral braniti razvojno skupino pred zunanjimi vplivi in pritiski. Tega ni bilo.

Po teoriji agilnega pristopa je celoten Scrum tim lociran na eni lokaciji in dnevno sodeluje. Tega v našem primeru nismo upoštevali (slika 14) in glede na naravo dela ni bilo možno. Lahko bi optimizirali lokacijo in sicer tako, da bi bila skrbnik metodologije in skrbnik izdelka za čas projekta v Ljubljani. Vendar bi bilo to povezano z dodatnimi visokimi stroški, ki niso bili planirani na začetku projekta.

3.7 Ostale težave

3.7.1 Spremembe v vodstveni strukturi pred začetkom in med projektom

Dolgoletni vodja našega oddelka lociran v Nemčiji, ki je imel tudi programersko ozadje je zagovarjal argumente za SQL podatkovno bazo, s čimer se napredni tehnični vodja oddelka v Sloveniji ni strinjal. Naslednik vodje našega oddelka lociran v Nemčiji ni imel programerskega ozadja, a je bil dober pogajalec in odprt za nove načine in razvoj. Tako smo v oddelku dobili sogovornika in vodjo, ki nas je bil pripravljen podpreti pri zahtevnem projektu, ki smo ga želeli izpeljat. Med prvo iteracijo smo dobili novega vodjo oddelka in novega vodjo projekta s sedežem v Milton Keynesu v Angliji.

Menjava vodij, projektnega managerja oz. skrbnika metodologije v okviru projekta ni skladno z dobro poslovno prakso, ob tem, da ima vsaka menjava vsebinske in stroškovne posledice, ne glede na metodologijo, ki se jo uporablja.

3.7.2 Širitev razvojnega tima z dvema dodatnima programerjema

Enega smo kot razvojni tim lahko sami izbrali in se je odlično vklopil v obstoječo skupino. Poleg tega je imel programerska znanja, ki je naši skupini manjkala.

Drugega je izbral vodja oddelka v Nemčiji. Kriterij je bil, da govori turško, saj je naša stranka v Turčiji ves čas želela človeka, ki bi znal lokalni jezik in bi bilo za definiranje zahtevkov bolj enostavno. Stranka v Turčiji je izkazala željo po tem, da je prisoten v Turčiji, tako, da sploh nismo imeli možnosti spoznati njega in njegove sposobnosti. Po dveh mesecih ni bilo nobene dodane vrednosti. Zahtevki niso bili nič bolj razumljivi, ker ni imel potrebnega poslovnega znanja in ni poznal funkcionalnosti obstoječe aplikacije,

znanja si ni mogel nabrati, ker ni bil vsakodnevno prisoten v razvojnem timu. Po dveh mesecih je zapustil podjetje. Tako smo izgubljali dragocen čas in denar namenjen projektu, z menjavo oz. iskanjem primernih kadrov.

Kasneje smo ugotovili, da bi potrebovali vsaj sedem ali osem novih članov razvojnega tima, da bi se lahko držali vseh rokov in da bi dostavili programsko opremo z dogovorjenimi funkcionalnostmi v dogovorjenem roku. Planiran proračun ne bi vzdržal. Sploh zaradi zahtev, da so člani razvojnega tima locirani v Turčiji.

3.7.3 Motiv razvojnega tima in naročnika

Razvojni tim v Sloveniji je imel profesionalen pristop in velik interes za uspešen zaključek projekta, saj bi si s tem zagotovili nadaljnji obseg dela v podjetju. Ob tem bi si zagotovil tudi možnost kandidiranja za naslednje projekte v koncernu. Tehnični aspekti projekta so bili zelo zahtevni. Razvojni tim je naredil izredno hiter preskok naprej in je deloval zelo predano projektu. Bil je kljub nezadovoljstvu končnega uporabnika zelo motiviran, saj so bili rezultati tehnične nadgradnje zelo dobri in uporabni. A za končnega uporabnika nevidni in s tem zaradi nestrokovnega ocenjevanja nič vredni. Naš razvojni tim je bil managementu predstavljen kot počasen in nezmožen. Management je na osnovi dolgotrajnih zahtev naročnika sprejel poslovno odločitev, da se projekt prekine

Naročnik je bil primoran sodelovat pri projektu, čeprav je bila njegova vizija o novi, izboljšani programski opremi drugačna. Naročnik je namreč želel lokalni turški razvojni tim. Management v Turčiji in v Nemčiji nista bila usklajena, med njima so bila celo trenja in nezaupanje. Izkazalo se je, da so bili pogledi na strategijo neenotni in obstajal je močan interes turških uporabnikov za že izbranega lokalnega razvijalca.

V tej točki ugotavljam, da mora motiv sodelovanja izhajati iz naročnika, saj je tako možnost uspešnega zaključka projekta bolj verjetna.

Dodatno smo imeli oteženo delo, saj v koncernu v tistem času niso bili naklonjeni agilnemu pristopu, saj še niso imeli znanja in vpogleda v ta novejši pristop. Koncern tudi še ni imel interne metodologije, kateri bi sledili in tako smo delovali kot posebneži.

3.8 Predlogi za naslednje projekte

Agilni pristop ne daje dovoljenja za nekontrolirano spreminjanje vsebine ciklov! S tehničnega vidika je imel razvojni tim zelo dobro razumevanje projekta. Slabši vpogled v projekt je imel le skrbnik metodologije, kot tudi skrbnik izdelka. Zaradi tega smo se soočili s kratkoročnimi spremembami obsega prvega cikla, kar je za agilni pristop nesprejemljivo. Saj se po metodologiji zahteva, da se dodajo dodatni zahtevki le, ko razvojni tim že vse naredi in sam izbere dodatne zahtevke (glede na prioritetno listo), ki jih lahko še izdela. Razmišljanje skrbnika izdelka, da mu agilni pristop omogoča sprotno prilagoditev končnega izdelka, ki bo vseboval vse njegove želje je nesprejemljivo in zmotno. Cikel je točno določen in nespremenljiv. Na koncu vsakega cikla se stranki izroči izdelek, ki ima dogovorjeno uporabno vrednost.

Količina dela naj bo v obsegu zmogljivosti razvojnega tima! Prvi cikel je bil glede na pravila agilnega pristopa definitivno predolg in je vključeval preveč tehničnih sprememb in nadgradenj, ki jih ni bilo mogoče izročit in uporabit in s tem za končnega uporabnika niso imele uporabne vrednosti.

Lokacija Scrum tima mora biti na enem mestu! Poslovno okolje je bilo izredno zahtevno. Delovali smo na različnih lokacijah, pravzaprav različnih državah in tudi v različnem kulturnem okolju. Dodatno je delo oteževalo tudi, da smo delovali v treh različnih časovnih conah z zamikom po eno uro. Turški uporabniki so bili zaradi izrednega povečevanja obsega dela preobremenjeni z delovnimi zadolžitvami. Vsako dodatno delovanje pri razvoju nove programske opreme je bilo veliko dodatno breme. Prepreka je bila poleg vsega naštetega še stopnja znanja angleškega jezika, ki je pogosto onemogočala kakovostno komuniciranje.

Podjetje mora imeti projekt v strateškem planu! Evropski projektni odbor ima glavno besedo pri projektih v podjetju. V projektnem odboru delujejo direktorji Evropskih hčerinskih podjetij. V projektnem odboru ni bilo tehničnega kadra. V letu 2012 se je začel tudi vseevropski projekt GET ONE, katerega cilj je poenotiti vso programsko opremo v evropskih državah. Za začetek v desetih, ki bi imele vlogo pilota. S tem je naš projekt dobil sekundarno vlogo na projektnem odboru, saj so planirali, da bo projekt GET ONE poskrbel tudi za turški trg.

Naročnik mora imeti potrebo po izvajanju projekta! Človeški viri so bili v našem primeru skrbnik izdelka, skrbnik metodologije in razvojni tim. Razvojni tim je bil strokovno podkovan in zelo motiviran, saj bi si z uspešnim projektom zagotovili tudi delo v prihodnje. Naloga, ki smo si jo zadali je bila zelo obsežna, a s strani skrbnika izdelka nismo imeli primerne podpore. Pravzaprav se je izkazalo, da skrbnik izdelka sploh ni zainteresiran za uspešnost projekta. Poleg tega, pa ni bil dovolj strokoven in kompetenten za dodeljeno delo. Skrbnik metodologije je bil prvič v tej vlogi. Scrum pristopa ni poznal. Nekako je deloval v vlogi klasičnega projektnega managerja, ki je sledil Houston pristopu.

Po analizi sem ugotovila, da kombiniranje slapovnega in agilnega pristopa ni problematično, saj imata oba sistematično razvite faze oz. področja, katera olajšajo delo na projektu, če jim slediš. Nikjer nisem ugotovila, da bi bilo v nekem trenutku bolje uporabiti slapovni pristop ali obratno. Preveč površno smo uporabljali tako en, kot drug pristop. Pomembno je, se držati pravil enega ali drugega pristopa. Mislim le, da je največja napaka neupoštevanje dejstva, da je naročnik tisti, ki mora dati projektu zagon.

SKLEP

Projekti so sestavljeni iz več faz. Pogosto lahko govorimo o fazi snovanja, kjer se porodi ideja, o fazi planiranja kjer definiramo, kaj bomo delali, o fazi izvedbe, kjer izdelujemo izdelek, ga testiramo vključujemo spremembe in ravnamo s tveganji in nazadnje s fazo zaključka, kjer projekt zaključimo, pripravimo potrebno dokumentacijo za naročnika in

pripravimo tudi seznam spoznanj in izkušenj, ki nam bodo v pomoč ob naslednjih projektih.

Za razvoj programske opreme se uporablja več različnih pristopov. Starejši je slapovni pristop, katerega značilnosti so, da so vse faze projekta zaporedne. V novejšem času pa se za razvoj programske opreme pogosto uporablja agilni pristop, saj zahteve današnjega sveta zahtevajo hitre rešitve in izboljšano programsko opremo takoj. Vsak od njiju ima prednosti in slabosti, pogosto pa se širi mnenje, da je agilni pristop boljši za naročnika in razvojni tim, ker ni potrebno specificirati zahtev, kar pa ni res, saj brez specifikacij tudi razvojni tim ne more izdelati nove programske opreme.

Razlike med slapovnim in agilnim pristopom so brez analize velike, ko pa se poglobimo v dejansko delo, ki ga je potrebno opraviti, spoznamo, da so dela približno enaka, le opravijo se v različnih fazah in na drugačen način. Najočitnejša razlika je seznam zahtevkov, ki se pri slapovnem natančno pripravi še pred začetkom projekta, medtem, ko se pri agilnem pripravi grob seznam. Pri agilnem se vsaka faza projekta večkrat ponovi. In s tem naročnik oz. uporabnik hitreje dobi delne rešitve. Če pogledamo celotno izvedbo, se bo končni izdelek najverjetneje razlikoval, saj naj pri slapovnem ne bi bilo odstopanj od začetne specifikacije, pri agilnem, pa se sproti vključuje spremembe in prednostne zahteve.

Projekt, ki sem ga analizirala je bila kombinacija Houston slapovnega pristopa in agilnega pristopa. Imeli smo seznam zahtev, a ne točno specificirane. Ravnali smo se po področjih znanja in imeli tedenske sestanke za pregled statusa le-teh, kar je v nasprotju z agilnim pristopom. Lokacija udeležencev ni bila na enem mestu. Nismo imeli klasičnega dnevnega 15 – minutnega sestanka, kot ga predpisuje Scrum. Razvojni tim je bil motiviran za uspešen zaključek projekta, saj bi si s tem zagotovil delo še v prihodnje. Projektni manager oz. skrbnik metodologije svojih nalog ni v celoti opravljal niti po slapovnem pristopu niti po agilnem pristopu. Skrbnik izdelka ni imel interesa uspešno zaključit projekt.

Analiziran projekt ima svoje prednosti in slabosti, ki sem jih zapisala v tretjem poglavju. Projekt ni bil uspešno zaključen iz več razlogov. Z mojega vidika ni bila težava v kombiniranju pristopov in neupoštevanju vseh pravil slapovnega Houston pristopa ali scrum pristopa. Menim, da je bil neuspešen, ker ni bil naročnik tisti, ki si je želel tega izdelka. Poleg tega se je spremenila še strategija podjetja in na koncu smo bili le v razvojnem timu zainteresirani za zaključek projekta. Naša naloga je bila zagotoviti nemoteno delovanje programske opreme. Naročnik si je pa želel popolnoma novo, drugačno in kar je najbolj pripomoglo k neuspešnosti projekta že vnaprej izbrano lokalno različico programske opreme. Proti toku je dolgoročno težko vzdržat.

Predvsem pa se mi je ob analiziranju pristopov in pisanju diplomskega dela postavilo vprašanje, ali so pristopi dejansko tako različni, da jih je treba obravnavat kot različne pristope, ali je to le kombiniranje pristopov. Morda je to celo optimizacija obstoječega slapovnega pristopa. Je agilnost noviteta in dejansko nekaj povsem drugega? Bomo v prihodnosti vse projekte delali agilno?

Opozarjam, da tematika še ni dorečena. V mojem primeru sem težko opredelila kam spadajo vsi koraki v našem projektu. Kako opredeliti kontrolne mejnike in odločevalne mejnike po agilni metodi, saj menim, da uprava podjetja, ki je naročnik in uprava podjetja, ki je izvajalec, potrebujeta določeno kontrolo nad potekom projektov. Mislim, da je ostalo še nekaj tematike, katerih vsebine bi bile lahko tema nadaljnjih raziskovanj in analiz.

Izpostavila bi še, da je z mojega vidika najbolj pomembno, da ima naročnik oz. uporabnik največji interes, da čim prej dobi zanesljivo delujočo programsko opremo. Brandonovo načelo agilnega pristopa, da naj se uporablja le tehnike, katere se obvlada, ima tudi velik smisel in bi ga bili morali upoštevat pri našem projektu.

LITERATURA IN VIRI

1. Beck, K., Beedle, M., Bennekum, A., Cockburn, A.,Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001, 11. februar). Agile manifesto. Najdeno 14. marca 2013 na spletnem naslovu http://agilemanifesto.org/
2. Brandon, D. (2006). *Project Management for Modern Information Systems.* Hershey: IRM Press.
3. Cohn, M. (2014, 6. junij) Scrum Iteration model. Najdeno 20. februarja 2016 na spletnem naslovu: https://www.mountaingoatsoftware.com/agile/scrum/images
4. Choudhury, A. (2012, 12. marec). Agile Vs Waterfall. Najdeno 20. februarja 2016 na spletnem naslovu http://www.sdlc.ws/agile-vs-waterfall
5. Daimler AG. (2011a). *Houston Project Guide V2.2.* (interno gradivo). Stuttgart: Daimler AG.
6. Daimler AG (2011b). *Governance Guideline.* (interno gradivo). Stuttgart: Daimler AG.
7. Daimler AG (2012). *Product backlog.* Stuttgart: Daimler AG.
8. Duggal, J. S. (2010, 9. julij). How Do You Measure Project Success? Rethinking the Triple Constraint. *PMI Community Post*. Najdeno 18. februarja 2013 na spletnem naslovu http://www.pmi.org/Knowledge-Center/Next-Level-Up-How-Do-You-Measure-Project-Success.aspx
9. Gladkova, E. (2012, 22. avgust). Agile vs Waterfall Development model. Najdeno 1. marca 2016 na spletnem naslovu http://www.sysgears.com/articles/agile-vs-waterfall-development-model/
10. Highsmith, J. (2002), *Agile Software Development Ecosystems.* Boston: Addison Wesley.
11. Kerzner, H. (2009). *Project Management A System Approach To Planning, Scheduling and Controlling* (10th ed.). New York: John Wiley & Sons.
12. May, E. L. & Zimmer, B. A. (1996) Evolutionary development. *Hewlett-Packard Journal*. August 1996, stran 1. Najdeno 10. marca 2013 na spletnem naslovu http://www.hpl.hp.com/hpjournal/96aug/aug96a4.pdf
13. Mieritz, L. (2012, 1. junij). Gartner Survey Shows Why Projects Fail. Najdeno 17. novembra 2013 na spletnem naslovu http://thisiswhatgoodlookslike.com/2012/06/10/gartner-survey-shows-why-projects-fail/

14. Greene, J. & Stellman, A. (2007). *Head First PMP*. Sebastopol: O'Reilly.
15. PMBOK – A guide to the project management body of knowledge. (2008). Newtown Square: Project management institute.
16. Poslovna aplikacija. (b.l.). V *iSlovarju*. Najdeno 18. februarja 2013 na spletni strani http://www.islovar.org/iskanje_enostavno.asp?reset=true
17. Rowley, J. (2013, 10. januar). 5th Edition PMBOK Guide – Chapter 1: Project Constraints. Najdeno 17. novembra 2013 na spletnem naslovu http://4squareviews.com/2013/01/10/5th-edition-pmbok-guide-chapter-1-project-constraints/
18. Schwaber, K., & Sutherland, J. (2012). Software in 30 days: How Agile Managers Beat the Odds, Delight Their Customers, And Leave Competitiors In the Dust. Hoboken: Wiley & Sons.
19. Schwaber, K., & Sutherland, J. (2013). *The Scrum Guide ™*. Najdeno 01. marca 2016 na spletnem naslovu http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf#zoom=100
20. Stare, A. (2010, 15. december), Vloga in naloge managerja projekta. *Projektni management*. Najdeno 1. marca 2016 na spletnem naslovu https://projektni-management.si/2010/12/15/vloga-in-naloge-managerja-projekta/
21. Stare, A. (2011). *Projektni management: teorija in praksa.* Ljubljana: Agencija Poti.
22. Stellman, A. (2010, 23. september). A Few Things Every Job-Seeking Programmer Should Know About Project Management (part2). Najdeno 1. marca 2016 na spletni strani http://www.stellman-greene.com/2010/09/23/a-few-things-every-job-seeking-programmer-should-know-about-project-management-part-2/
23. Spletna aplikacija (b.l.). V *Slovar slovenskega knjižnega jezika*. Najdeno 18. februarja 2013 na spletnem naslovu http://bos.zrc-sazu.si/sskj.html
24. Veethil, S. T. (2013, 3. maj) Risk Management in Agile. *Scrum Alliance*. Najdeno 15. marca 2016 na spletnem naslovu https://www.scrumalliance.org/community/articles/2013/2013-may/risk-management-in-agile
25. Viscardi, S. (2013). *The Professional ScrumMaster's Handbook.* Birmingham: Packt Publishing. Najdeno 01. marca 2016 na spletnem naslovu https://www.safaribooksonline.com/library/view/the-professional-scrummasters/9781849688024/ch01s04.html

26. Wysocki, R.K. (2006). *Effective Software Project Management.* Indianapolis: Wiley Publishing.