UNIVERZA V LJUBLJANI

EKONOMSKA FAKULTETA

MAGISTRSKO DELO

MANAGEMENT ČASA IN KAKOVOSTI PROJEKTOV RAZVOJA PROGRAMSKE OPREME: PRIMER IZBRANEGA PODJETJA

Ljubljana, september 2013 DEJAN VATOVEC

IZJAVA O AVTORSTVU

Spodaj podpisani Dejan Vatovec, študent Ekonomske fakultete Univerze v Ljubljani, izjavljam, da sem avtor magistrskega dela z naslovom MANAGEMENT ČASA IN KAKOVOSTI PROJEKTOV RAZVOJA PROGRAMSKE OPREME: PRIMER IZBRANEGA PODJETJA, pripravljenega v sodelovanju s svetovalcem prof. dr. Talibom Damijem.

Izrecno izjavljam, da v skladu z določili Zakona o avtorskih in sorodnih pravicah (Ur. l. RS, št. 21/1995 s spremembami) dovolim objavo magistrskega 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 oziroma avtoric, ki jih uporabljam v magistrskem delu, citirana oziroma 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, št. 55/2008 s spremembami);
- se zavedam posledic, ki bi jih na osnovi predloženega magistrskega dela dokazano plagiatorstvo lahko predstavljalo za moj status na Ekonomski fakulteti Univerze v Ljubljani v skladu z relevantnim pravilnikom.

V Ljubljani, dne 30. septembra 2013 Podpis avtorja:_____

KAZALO

UVOD **1**

1 MANAGEMENT PROJEKTOV **3**

1.1 Življenjski cikel projekta 4

1.2 Področja projektnega managementa 7

2 MANAGEMENT ČASA **9**

2.1 Opredelitev aktivnosti 11

2.2 Razvrščanje aktivnosti 11

2.3 Ocenjevanje potrebnih virov 14

2.4 Ocenjevanje trajanja aktivnosti 15

2.5 Izdelava časovnice projekta 16

2.6 Kontrola poteka projekta 21

3 MANAGEMENT KAKOVOSTI **26**

3.1 Opredelitev in pristopi k definiciji kakovosti 27

3.2 Planiranje kakovosti 27

3.2.1 Vhodi v proces planiranja kakovosti 27

3.2.2 Tehnike planiranja kakovosti 28

3.2.3 Izhodi iz procesa planiranja kakovosti 28

3.3 Zagotavljanje kakovosti 29

3.3.1 Vhodi v proces zagotavljanja kakovosti 30

3.3.2 Orodja in tehnike zagotavljanja kakovosti 30

3.3.3 Izhodi iz procesa zagotavljanja kakovosti 30

3.4 Kontrola kakovosti 30

3.4.1 Orodja za kontrolo kakovosti 31

3.4.2 Izhodi iz procesa kontrole kakovosti 31

4 KAKOVOST PROGRAMSKE OPREME **32**

4.1 Definicija kakovosti programske opreme 32

4.2 McCallov model kakovosti programske opreme 32

4.2.1 Skupina operativnosti programske opreme 32

4.2.2 Skupina vzdrževalnosti programske opreme 33

4.2.3 Skupina prenosljivosti programske opreme 34

4.3 Standardi kakovosti s področja programske opreme 34

4.3.1 ISO 9126 36

4.3.2 ISO/IEC 12207 37

4.3.3 IEEE 829 ... 38
4.3.4 IEEE 1016 ... 38
5 RAZVOJ PROGRAMSKE OPREME ... 39
5.1 Opredelitev programske opreme ... 40
5.2 Procesi specifični razvoju programske opreme ... 41
5.3 Modeli razvoja programske opreme ... 42
5.4 Zagotavljanje in kontrola kakovosti programske opreme ... 44
6 TESTIRANJE PROGRAMSKE OPREME ... 47
6.1 Testiranje kot orodje zagotavljanja in kontrole kakovosti ... 47
6.2 Proces testiranja ... 48
6.2.1 Priprava testiranja ... 48
6.2.2 Izvedba testiranja ... 50
6.2.3 Predmet testiranja ... 53
6.2.4 Zaključevanje testiranja ... 54
6.3 Vrste testiranja ... 54
6.4 Nivoji testiranja ... 55
7 ANALIZA IZBRANEGA PODJETJA ... 56
7.1 Predstavitev podjetja ... 58
7.2 Model CMMI ... 59
7.3 Identifikacija pomanjkljivosti ... 61
7.3.1 CMMI analiza ... 61
7.3.2 Model razvojnega procesa ... 64
7.4 Ukrepi za odpravo pomanjkljivosti ... 64
7.4.1 CMMI analiza sprememb ... 65
7.4.2 Uvedba agilnega modela razvoja programske opreme ... 68
7.4.3 Uvedba aktivnosti za izboljšanje kakovosti programske opreme ... 70
7.5 Model TMMI ... 73
7.5.1 Analiza procesa testiranja ... 74
7.5.2 Uvedba sprememb v proces testiranja ... 75
7.6 Predlogi nadaljnjih izboljšav ... 77
SKLEP ... 78
LITERATURA IN VIRI ... 81

KAZALO SLIK

Slika 1: Trojna omejitev ciljev projekta 4
Slika 2: Življenjski cikel projekta s prikazom intenzitete aktivnosti 5
Slika 3: AOA diagram projekta 12
Slika 4: PDM diagram projekta 13
Slika 5: Tipi povezav med aktivnostmi projekta 13
Slika 6: Primer Gantt diagrama s pomeni elementov 17
Slika 7: Kritična pot projekta 18
Slika 8: Prikaz delovanja metode prislužene časovnice 25
Slika 9: Kategorije ter pod-kategorije kakovosti po standardu ISO 9126 37
Slika 10: Naraščajoči stroški hrošča skozi faze razvoja programske opreme 47
Slika 11: Krivulja intenzivnosti testiranja v posameznih razvojnih fazah 50
Slika 12: Prikaz strategij testiranja »od spodaj-navzgor« ter »od zgoraj-navzdol« 51
Slika 13: TDD pristop k razvoju programskih modulov 71
Slika 14: Lestvica ocenjevanja kakovosti 72
Slika 15: Življenjski cikel programskega hrošča 76

UVOD

V magistrskem delu želim na praktičnem primeru prikazati management časa ter kakovosti projektov razvoja programske opreme na področju vgrajenih sistemov. Praktično izvajanje bo temeljilo na teoretičnih osnovah, na katerih bo narejena analiza projektnega managementa in končna ocena celotnega procesa s poudarkom na pridobljenih izkušnjah za prihodnje projekte.

Management časa ter management kakovosti se najbolj odrazita na izdelku, ki ga ponudimo trgu. Uspešen management časa neposredno vpliva na potreben čas, da je izdelek na trgu. Upravljanje s kakovostjo pa neposredno vpliva na lastnosti samega izdelka, tako z vidika njegove tehnične kakovosti, kot tudi z vidika kakovosti oz. kako dobro izdelek zadovoljuje potrebe in želje odjemalcev.

Namen magistrskega dela oz. razlog za njegov nastanek so zaznane težave pri razvoju novih produktov, predvsem težave s kakovostjo ter rabo časa, ki ima posreden vpliv tudi na kakovost.

Kakovost izdelka moramo planirati, zagotavljati ter kontrolirati skozi vse faze njegovega razvoja. Zagotavljanje ter kontrolo v praksi zagotovimo predvsem s testiranjem, zato bo področje testiranja programske opreme podrobneje obravnavano.

Na drugi strani pa je potrebno izdelek tudi v najkrajšem možnem času ponuditi trgu. Kar najvišja kakovost v najkrajšem možnem času sta si nasprotujoča si cilja. V nalogi je cilj raziskati ter izvesti najboljše prakse tovrstne problematike na praktičnem primeru in izboljšati izvajanje dela v razvojnem oddelku obravnavanega podjetja ter dosegati hitrejše in učinkovitejše izvajanje razvojnih nalog, kar pripomore k hitrejšemu nastopu izdelka na trgu in hkrati ponuditi kakovostnejši produkt.

Vse to je potrebno zaradi sprememb, ki smo jim priča v delovanju sodobne družbe in tudi v poslovnem okolju. Spremembe silijo podjetja v visoko stopnjo prilagodljivosti na področju ponudbe izdelkov in storitev. Tovrstno prilagodljivost lahko podjetja dosežejo le s primerno organiziranostjo, strateško usmeritvijo, delovanjem navzven ter predvsem navznoter z aktivnostmi, kot so izkoriščanje in nadgradnja svojih prednosti ter z izboljšavami na področjih, ki se jih zavedajo kot svoje slabosti.

Mnoga podjetja te cilje dosežejo z organiziranjem svojega delovanja v projekte, kar posledično vpliva na potrebo po razvoju projektnega managementa kot samostojne discipline znotraj podjetja (Šajtegel, 2003, str. 94). Projektna usmerjenost in njeno obvladovanje, omogočata podjetjem, da učinkovito izkoristijo svoje zaposlene ter njihova znanja z namenom, da bi izdelke kar najhitreje ponudila trgu in na najvišjem kakovostnem nivoju.

V projektno organiziranih podjetjih vloga projektnih managerjev in njihove kompetence s področij projektnega managementa odločata o uspehu ali neuspehu projektov.

Projektni pristop pomeni delo v dinamičnem okolju, izraža potrebo po visoki učinkovitosti, zahteva optimalno rabo kadrov in hiter vpogled v tekoče stanje del na projektu (Kampuš, 2002, str. 47). Projektni managerji morajo, zaradi kompleksnosti naštetega, znati uporabljati projektnemu managementu namenjena informacijska orodja, ki so pomemben del projektnega managementa v fazi planiranja kot tudi v fazi izvedbe in kontrole.

Management časa je ključna naloga projektnega managerja in je odločilnega pomena za uspešno končanje projekta. Faza planiranja časa zahteva tako tehnična znanja s področja, kot tudi predhodne izkušnje na podobnih projektih. Za uspešno planiranje časa je potrebno obravnavati več glavnih procesov (Project Management Institute, 2004, str. 129, v nadaljevanju PMI), od identifikacije nalog, ki so potrebne za končanje projekta, vse do ocenjevanja potrebnega časa za končanje posameznih nalog.

Naloga projektnega managerja je tudi skrb za ustrezno kakovost produkta. Slabo planiranje kakovosti je v neposredni povezavi z uspehom ali neuspehom projekta (Fuller, Wallach & George, 2008, str. 282). Zanemarjanje vidika kakovosti ima za posledico produkt slabe kakovosti, ki je praviloma na trgu neuspešen. Zaradi reklamacij pa povzroči podjetjem nenačrtovane in nepotrebne stroške, tako s finančnega, kot tudi s časovnega vidika. Popravilo ali vzdrževanje produktov slabe kakovosti zasedata vire razvojnih ekip, ki so že razporejeni na nove naloge.

Kakovost je torej potrebno vnaprej planirati, zagotavljati ter kontrolirati skozi vse faze projekta (Schwalbe, 2010, str. 292–295). Tehnike za zagotavljanje kakovosti so različne, najpogosteje pa je kakovost zagotovljena in kontrolirana preko testiranja produkta, praviloma v vseh razvojnih fazah. Zagotavljanje kakovosti na tak način, se odrazi tudi v planiranju časa, v katerem je del aktivnosti namenjen testiranju.

Predpogoj za uspešnost projekta je odlično opravljeno delo projektnega managerja že v začetni fazi projekta. Slabo opravljeno delo v začetni fazi, se bo negativno odrazilo v fazi izvedbe. Težave se bodo praviloma potencirale. Slabo planiranje kasneje pomeni slabo kontrolo (Lewis, 1995, str. 18).

Problematika, ki je v magistrskem delu obravnavana, torej obvladovanje časa in kakovosti ter tudi problematika nekaterih drugih področij projektnega managementa, je ključnega pomena za uspeh projekta. Obvladovanje časa se odrazi v pravočasni izvedbi projektov, obvladovanje kakovosti pa v skladnosti produkta s pričakovanji odjemalca. Obe področji sta medsebojno povezani, zato je usklajenost obvladovanja obeh ključnega pomena (Vrhovšek, 2005, str. 93). Obvladovanje obeh področij se kaže v uspehu podjetja. Projekti, ki so uspešni, so končani v predvidenih rokih, njihovi stroški niso preseženi in produkt je ustrezne kakovosti.

V magistrskem delu identificiram probleme iz realnih oz. praktičnih primerov projektov razvoja programske opreme. Obravnava je izvedena v manjši razvojni ekipi, ki šteje le nekaj članov.

Produkti, ki jih izbrano podjetje razvija in tudi proizvaja, so s področja vgrajenih sistemov in njihov razvoj vključuje tako razvoj strojne kot tudi programske opreme. V magistrskem delu se posebej posvečam obravnavi ter analizi razvoja programske opreme.

Pred analizo je bilo potrebno podrobno preučiti teorijo s področja projektnega managementa in s področja managementa razvoja programske opreme. Za preučevanje je bila uporabljena domača ter predvsem tuja strokovna literatura, elektronski viri in v dobršni meri tudi tehnična in organizacijska znanja ter izkušnje, ki sem jih pridobil med študijem in med delom na različnih projektih s področja razvoja programske opreme v podjetjih, kjer sem deloval.

Delo vključuje preučevanje standardov s področja managementa kakovosti ter uporabo primerov najboljših praks. Delovanje podjetja primerjam s teoretičnimi primeri in standardi ter identificiram največja odstopanja, ki negativno vplivajo na delovanje procesa. Za odstopanja, na osnovi raziskave ter analize, podajam predloge za izboljšave na področjih upravljanja s časom ter kakovostjo oz. tudi na področjih, ki so v korelaciji z omenjenima. Obenem iščem tudi delovanje, ki je v skladu s priporočili in ga je potrebno ohraniti takega, kot je.

Poleg primerjave med teoretičnim znanjem in praktičnim izvajanjem iščem odgovore na identificirane probleme tudi preko mnenj in izkušenj drugih udeležencev projekta. Obravnavani so praktični problemi, s katerimi se udeleženci srečujejo, in načini, kako jih rešujejo.

Praktični del naloge bi opredelil za empiričnega, saj bazira na opazovanju dogodkov razvojnega procesa. Merljivih podatkov, npr. o trajanju aktivnosti ali rabi časa v času raziskave ni ali jih ni možno primerjati s podatki pred uvedenimi spremembami. Eden izmed ciljev raziskave ter uvedbe sprememb, je tudi objektivno merjenje lastnosti razvojnega procesa, kar se tiče produktivnosti, učinkovitosti, kakovosti ipd.

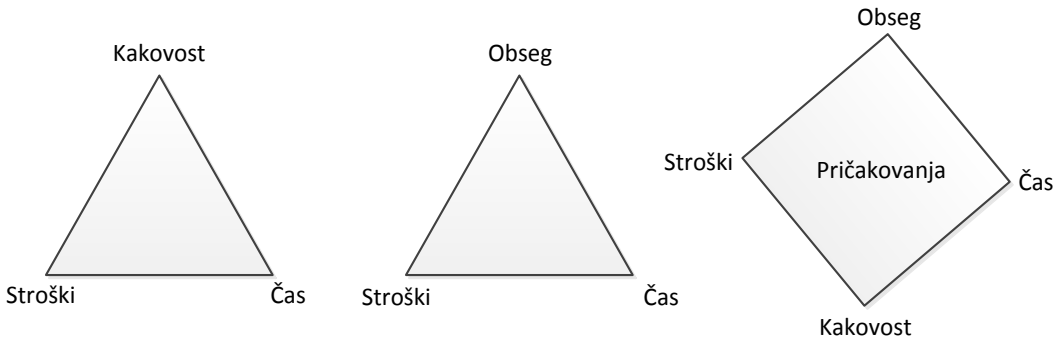
1 MANAGEMENT PROJEKTOV

Izraz management projektov pomeni aplikacijo znanj, orodij in tehnik z različnih področij s ciljem uspešno končati projekt oz. njegov produkt in se vrši skozi vse faze življenjskega cikla projekta. Faze projekta so zagon, faza planiranja, izvedbe, spremljanja in kontrole ter zaključevanja. Za izvajanje faz je odgovorna oseba, projektni manager, ki mora težiti h končanju projekta v predvidenem obsegu, znotraj časovnih in stroškovnih okvirjev, ter hkrati težiti k zadovoljevanju pričakovanj deležnikov projekta (Schwalbe, 2010, str. 10).

Projekt opredelimo kot začasen podvig, katerega namen je unikaten izdelek ali storitev. Projekt je začasen, ker ima svoj začetek, trajanje in konec. Vsak projekt se konča, bodisi uspešno, s končanim produktom, ali v trenutku, ko je jasno, da je pozitiven izid projekta nemogoče doseči ali zanj ni več potrebe in se posledično zaključi (PMI, 2004, str. 5).

Projektni management zajema aktivnosti, kot so identifikacija zahtev, postavitev ciljev, obravnavanje in balansiranje vsebinskih, časovnih in stroškovnih omejitev projekta ter prilagajanje produkta in drugih atributov projekta zahtevam deležnikov (PMI, 2004, str. 8). Ko govorimo o medsebojni povezavi omejitev oz. ciljev projekta, govorimo o trojni omejenosti obsega, časa ter stroškov oz. o četverni omejenosti, ko je kot parameter projekta enakovredno vključena še kakovost.

Slika 1: Trojna omejitev ciljev projekta



Vir: Povzeto po D. Lock, Project Management, 2007, str. 21 ; D. Haughey, Understanding the Project Management Triple Constraint, 2013.

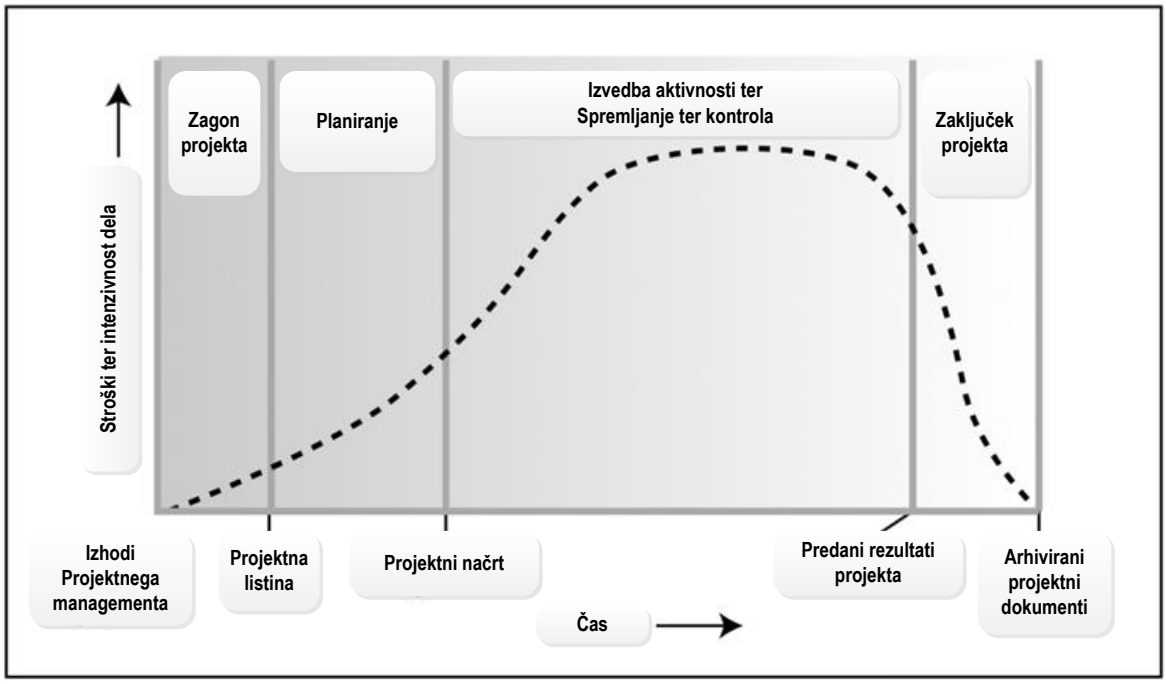
Sredi 80-ih let 20. stoletja je dr. Martin Barnes predstavil originalno verzijo trikotnika projektnih ciljev, ki jo kaže levi diagram. Želel je prikazati, da se bodo projektni managerji za dosego izbranega cilja morali odreči drugim ciljem (Lock, 2007, str. 21). V literaturi srečamo tudi drugačne diagrame, dva od mnogih sta prikazana desno od Barnesovega originala. Tretji je v obliki diamanta, doda kakovost kot enakovreden cilj in postavi pričakovanja uporabnikov v sredino (Haughey, 2013).

1.1 Življenjski cikel projekta

Projektni managerji projekt razdelijo v faze, ki predstavljajo življenjski cikel projekta. V literaturi so faze različno opredeljene in enotna delitev ne obstaja. V splošnem pa identificiramo zagon projekta, planiranje, izvajanje, spremljanje in kontrolo ter zaključevanje projekta. Faze povezujejo začetek projekta z njegovim koncem in so deljene glede na:

- vrste tehničnih nalog in opravil, ki jih je potrebno opraviti v posamezni fazi,
- čas oz. časovne roke, ko je potrebno predati predvidene rezultate projekta,
- potrebne vloge in potrebne vire za posamezne faze projekta,
- način izvedbe in kontrole posamezne faze.

Slika 2: Življenjski cikel projekta s prikazom intenzitete aktivnosti



Vir: Povzeto po PMI, A Guide to the Project Management Body of Knowledge, 2004, str. 16.

V fazi zagona določimo vsebino, obseg ter cilje projekta. Določimo datum začetka ter konca projekta ter razdelimo vloge na projektu, določimo torej projektni tim. Definicija projekta mora biti soglasna s strani vseh deležnikov projekta oz. vseh, ki jim je uspeh projekta v interesu (Vrhovšek, 2005, str. 10). Za projekt je ključna pravilna izbira projektnega managerja, ki bi bil idealno v projekt vključen že v fazi iniciacije oz. v morebitne aktivnosti pred njo. Vendar je večkrat projektni manager izbran šele, ko so mnoge odločitve v fazi zagona že sprejete (Schwalbe, 2010, str. 87).

Vhodi v fazo zagona so zamisel, kaj naj bo produkt projekta, strateški načrt, kriteriji za izbiro projekta ter informacije iz preteklosti. Izhodi so naročilo projekta, izbira projektnega managerja, zahteve, predpostavke projekta ter pogodbe (Vrhovšek, 2005, str. 13).

Planiranje projekta pomeni usklajevanje vseh treh oz. štirih soodvisnih omejitvenih ciljev, ko je v planiranje projekta oz. kot njen cilj pomembno vključena tudi kakovost (Schwalbe, 2010, str. 9).

Drugi primarni cilji projekta so:

- Obseg, ki pomeni količino dela, ki bo opravljeno na projektu in določa vsebino projekta oz. neposredno njegov izdelek ali storitev.
- Čas, ki pomeni trajanje projekta od njegovega začetka do njegovega konca. Management časa pomeni izdelavo in kontrolo časovnice, spremljanje izvedbe, korektivne ukrepe ter spremembe časovnice.

- Stroški projekta, ki pomenijo celotne stroške od začetka do konca projekta. Management stroškov pomeni planiranje proračuna projekta ter kontrolo porabe sredstev.

Faza planiranja vključuje načrtovanje obsega ter identifikacijo potrebnih aktivnosti za dosego ciljev projekta. Določimo način in pristop k izvajanju aktivnosti ter izvajalce aktivnosti. Identificiramo potrebna znanja in opremo za njihovo izvedbo in standarde, ki jim je potrebno zadostiti. Določimo odgovorne osebe za posamezne aktivnosti, določimo način izvedbe, stroške ter časovni okvir projekta (Vrhovšek, 2005, str. 13).

Projektni manager k planiranju pristopi od zgoraj navzdol ali od spodaj navzgor. Izbira je odvisna od narave projekta ter zahtevane točnosti ocenjevanja in nivoja definicije projekta. Prva metoda je primarno osredotočena na cilj in na večje skupine dela, ki jih je potrebno opraviti. V drugem pristopu najprej identificiramo vse aktivnosti projekta, ki se jih kasneje uvrsti v posamezne faze projekta. Aktivnostim pripišemo atribute časa, stroškov ter vsebino. S seštevkom vrednosti atributov, pridobimo informacijo o skupnem času, stroških ter vsebini projekta (Lock, 2007, str. 55).

Po določitvi parametrov posameznim aktivnostim, izvršimo razvrščanje (angl. *sequencing*) aktivnosti. Določimo vrstni red izvajanja aktivnosti, da bi dosegli najoptimalnejši potek projekta. Določimo tudi povezave med aktivnostmi oz. identificiramo kako so posamezne aktivnosti medsebojno povezane in odvisne (Schwalbe, 2010, str. 111). Projektni manager se med planiranjem poslužuje orodij ter metod kot so strukturirana členitev dela (angl. *Work Breakdown Structure*, v nadaljevanju WBS), Gantt diagram, tehnika PERT (angl. *Project Evaluation and Review Technique*), metoda kritične poti (angl. *Critical Path Method*, v nadaljevanju CPM) ter z drugimi.

Faza izvedbe pomeni opravljanje del in nalog na posameznih aktivnostih ter koordinacijo medsebojno odvisnih aktivnosti. Naloga projektnega managerja je, preko koordiniranja, zagotavljati nemoteno in neprekinjeno delo izvajalcev aktivnosti. Produkt projekta je v osnovi rezultat te faze. Faza izvedbe je najintenzivnejša faza projekta in zahteva največ virov (Schwalbe, 2010, str. 106). Proces faze izvajanja projekta vsebuje 7 pod-procesov (Vrhovšek, 2005, str. 7):

- izvajanje projektnega načrta,
- zagotavljanje kakovosti procesov projekta in produkta projekta,
- razvoj projektne ekipe v smislu sodelovanja članov in povečanja znanja posameznikov,
- management komunikacij oz. poročanje o napredku projekta,
- izvajanje povpraševanja po virih potrebnih za izvedbo projektov,
- izvajanje izbire virov za projekt ter
- izvajanje administrativnega dela, kot je npr. management pravnih vidikov projekta.

Faza spremljanja in kontrole je proces merjenja napredovanja projekta proti njegovim ciljem, opazovanje odstopanj od načrta ter odločanje in izvajanje ukrepov, da bi potek projekta pripeljali nazaj na planirano pot. Fazo kontrole izvajamo skozi celoten življenjski cikel projekta (Schwalbe, 2010, str. 111) in temelji na, v fazi planiranja, pripravljenem načrtu projekta. Načrt predstavlja osnovo po kateri izvajamo projekt. Z meritvami napredovanja posameznih aktivnosti in primerjanja z načrtovanim napredkom projekta določamo status projekta. Ugotavljanje odstopanj od referenčnega načrta je naloga projektnega managerja. Projektni manager ugotavlja odstopanja glede obsega, časa in stroškov projekta ter tudi kakovosti.

V procesu kontrole je ključno pravočasno in ustrezno ukrepanje, ki temelji na objektivnem merjenju napredovanja projekta. Pravočasno ukrepanje je predpogoj za to, da bi bili ukrepi uspešni. Da bi se lahko ustrezno ukrepalo, pa je pomembno imeti realne ocene o napredku na posamezni aktivnosti. Preveč optimistično poročanje, h kateremu so tipično nagnjeni izvajalci aktivnosti, se v kasnejših fazah negativno odrazi v premilih, nepravilnih ali prepoznih ukrepih. Le realno poročanje povzroči nasproten učinek oz. ustrezno pravočasno ukrepanje.

Faza zaključevanja je zadnja faza v življenjskem ciklu projekta. Da bi bil projekt izveden znotraj časovnih okvirjev, je tudi to fazo potrebno izvesti pred predvidenim datumom konca projekta. Glavne aktivnosti v fazi zaključevanja projekta delimo na zunanje, notranje ter aktivnosti, ki se nanašajo na sam projekt (Vrhovšek, 2005, str. 16), in so:

- Predaja rezultatov projekta oz. produkta projekta naročniku ter pridobitev potrditve o ustreznosti rezultata ter morebitne dopolnitve, ki jih je še potrebno izvesti, da bi bil projekt uspešno zaključen.
- Končanje aktivnosti z vidika lastnega podjetja. Predaja zaključnih poročil posameznih aktivnosti, ureditev dokumentacije, prerazporeditev osebja ter opreme na druge projekte.
- Priprava povzetka projekta. Kaj se je pričakovalo in kaj je bilo doseženo. Povzetek morebitnih sprememb in njihova obrazložitev ter ocena uspešnosti projekta.

Rezultat projekta in projekt sam sta vedno znova unikatna, naj gre za izdelek ali storitev kot možna produkta projekta. Tudi projekt, katerega načrtovani rezultat je podoben rezultatu projekta iz preteklosti, vedno teče v drugačnih okoliščinah kot pretekli projekt.

1.2 Področja projektnega managementa

Poznamo 9 področij projektnega managementa, ki jih mora obvladati uspešen projektni manager. Od teh štejemo 4 v skupino temeljnih oz. primarnih in 4 v skupino pomožnih oz. sekundarnih (Schwalbe, 2010, str. 12). Področje, ki vsa področja poveže imenujemo management integracije (angl. *Integration Management*). Med temeljna področja projektnega managementa sodijo management obsega, časa, stroškov ter kakovosti.

Temeljna področja managementa projektov so neposredno povezana s ciljem, ki je npr. opredeljen z obsegom projekta ali časovno z rokom končanja projekta.

Management obsega

Management obsega je management vseh opravil oz. aktivnosti, ki jih je potrebno opraviti za dosego končnega cilja, ki je predvideni produkt projekta. 5 osnovnih procesov, ki jih vključuje management obsega, je zajem zahtev stranke, definiranje obsega, izdelava WBS strukture, potrditev obsega s strani deležnikov projekta ter kontrola obsega (Schwalbe, 2010, str. 178). Natančna definicija obsega je ključna za uspeh projekta, ker je osnova za ocenjevanje časa in stroškov ter rezervacijo virov potrebnih za izvedbo vseh aktivnosti. Predstavlja tudi referenco za merjenje in kontrolo napredovanja izvedbe. Definicija oz. opredelitev aktivnosti je že v zgodnji fazi osnova za delegiranje odgovornosti za posamezne aktivnosti.

Management časa

Management časa vključuje aktivnosti ter procese, katerih cilj je zagotoviti, da je projekt končan znotraj časovnih okvirjev. Obširnejša opredelitev managementa časa ter teme planiranja, izvajanja aktivnosti ter kontrole časa so opisane v nadaljevanju dela.

Management stroškov

Management stroškov je izvajanje aktivnosti, katerih namen je, da se projekt uspešno zaključi znotraj odobrenih stroškov in skladno s trojno ali četverno omejitvijo projektnih ciljev. Procesi, ki so del managementa stroškov, so:

- Proces ocenjevanja stroškov projekta, katerega glavni namen je analitičen dokaz, da je projekt poslovno upravičen. Proces je osnova za dosego cilja, končati projekt znotraj odobrenih stroškovnih okvirjev.
- Proces delitve sredstev za posamezne aktivnosti z namenom, da se zagotovi sredstva, za njihovo izvajanje.
- Kontrola stroškov oz. primerjanje nastalih dejanskih stroškov s predvidenimi in izvajanje korektivnih ukrepov. Prilagajanje ocene stroškov in porabe sredstev je ves čas trajanja projekta potrebno in pričakovano izvajati. Poleg porabe sredstev med projektom izvajamo tudi merjenje pridobljenih koristi glede na porabljena sredstva.

Management kakovosti

Management kakovosti je področje managementa projektov, katerega cilj je zagotoviti produkt skladen s pričakovanji deležnikov projekta. Obširnejša definicija kaj kakovost je, pristopi k njej oz. različni vidiki kakovosti ter management kakovosti so opisani v nadaljevanju dela.

Med sekundarna področja projektnega managementa štejemo management človeških virov, management komunikacije, management tveganj ter management oskrbovanja projekta. Pomožna področja managementa projektov so potrebna, da bi projekt pripeljali do ciljev, ki jih opredeljujejo primarna področja projektnega managementa.

Management integracije

Management integracije zajema aktivnosti, ki identificirajo, kombinirajo in združujejo različne procese in aktivnosti z ostalih področij managementa projektov. Procesi integracije so ključni za uspešno končanje projekta, saj povezujejo aktivnosti ter prilagajajo pričakovanja deležnikov projekta. Procesi, ki so del managementa integracije in jih izvaja projektni manager in njegova ekipa, so (PMI, 2004, str. 71):

- Razvoj projektne listine (angl. *Project Charter*), ki pomeni izdelavo dokumenta, ki formalno opredeli cilje in legitimira projekt.
- Razvoj načrta managementa projekta, ki pomeni definicijo akcij, ki so potrebne za koordinacijo drugih načrtov, ki so del projekta.
- Direkcija izvedbe projekta, ki pomeni izvedbo del, potrebnih, da bi se doseglo načrtovane cilje projekta.
- Spremljanje in kontrola dela na projektu, ki zajema sledenje aktivnostim, pregledovanje napredka ter izvedbo korektivnih akcij.
- Spremljanje in kontrola zahtev po spremembah projekta, ki vključuje bodisi potrjevanje ali zavrnitev sprememb. V primeru potrditve sprememb, proces zajema ponovno definicijo projektnih ciljev, spremembe organizacijskih struktur podjetja ter dokumentiranje vseh sprememb v ustrezno dokumentacijo.
- Zaključevanje projekta, ki pomeni končanje vseh aktivnosti na vseh področjih projektnega managementa in formalen zaključek projekta.

Management integracije je nesporno pomemben del projektnega managementa, ki vodi do jasne povezanosti projektnih aktivnosti, npr. v procesu planiranja ali izvedbe. Cilji projekta so si običajno nasprotujoči in da bi dosegli en cilj, je potrebno žrtvovati druge cilje oz. sklepati kompromise.

2 MANAGEMENT ČASA

Cilj projektnega managementa je doseči rezultate projekta znotraj vseh predvidenih omejitev oz. ciljev, ki so primarno obseg, čas, stroški ter ustrezna kakovost. Uspešnost projektov se zelo pogosto interpretira ravno skozi dosego rezultatov znotraj časovnih okvirjev. Zaključiti projekt v časovnem okvirju, zastavljenem ob zagonu ter med planiranjem, je po mnenju večine projektnih managerjev največji izziv in predstavlja najpogostejši razlog za konflikte ter druge težave med izvedbo, kot tudi po navideznem zaključku projekta (Schwalbe, 2010, str. 212). Končanje projekta znotraj časovnih okvirjev

ima za podjetje nedvomno pozitiven poslovni učinek, saj je produkt na voljo na trgu v načrtovanem trenutku. Podjetje pa se lahko nemudoma loti novih podvigov.

Učinkovit management časa pripomore, da se med potekom projekta ne srečujemo s projektnemu delu običajnimi težavami. Čas je potrebno dojeti kot vir, ki ga je ključno čim bolje izkoristiti in ima neposreden vpliv na učinkovitost dela. Po drugi strani je neizkoriščeni čas vir, ki je za vedno izgubljen (Young, 2007, str. 222).

Čas je v primerjavi z nekaterimi drugimi parametri projekta enostavno meriti. Je fizikalna veličina, ki ima lastnost, da teče ne glede na aktivnosti, ki se vršijo na projektu. Ta lastnost, med procesom managementa časa, projektnim managerjem, nemalokrat predstavlja težave. Šest procesov managementa časa, katerih cilj je zagotoviti, da bo projekt končan znotraj časovnih okvirjev (Schwalbe, 2010, str. 213), je:

1. Opredelitev aktivnosti, ki morajo biti izvedene med projektom, da projekt vodi v načrtovane rezultate. Aktivnosti oz. naloge so elementarne enote dela, ki so del strukture WBS in so rezultat členjenja celotnega dela. Vsaka enota dela ima, podobno kot celoten projekt, definiran potreben čas za končanje, predvidene stroške ter vire potrebne za izvedbo.
2. Definicija povezav ter odvisnosti med posameznimi aktivnostmi v smislu iskanja najoptimalnejšega zaporedja izvajanja. Govorimo o razvrščanju aktivnosti. Rezultat tega procesa je mrežni diagram aktivnosti.
3. Ocenjevanje potrebnih virov za končanje posameznih aktivnosti. Med vire štejemo človeške vire, opremo ter materialne vire. Rezultat procesa je seznam virov potrebnih za končanje aktivnosti.
4. Ocenjevanje časa, ki je potreben za končanje vsake posamezne aktivnosti ob znanih drugih vplivnih parametrih. Rezultat procesa ocenjevanja je seznam potrebnega časa za končanje aktivnosti.
5. Razvoj terminskega plana oz. časovnice projekta, ki vključuje analizo zaporedij aktivnosti, ocenjevanje potrebnih virov ter ocenjevanje trajanja aktivnosti. Razvoj časovnice je kompleksen proces, ki je končan v več iteracijah. Običajno projektni managerji uporabijo računalniška orodja namenjena managementu projektov. Rezultat procesa je časovnica projekta, tipično predstavljena grafično kot Gantt diagram.
6. Kontrola časovnice, ki vključuje kontrolo ter management aktivnosti in se odraža v prilagajanju oz. spremembah časovnice ter v izvajanju korektivnih ukrepov z namenom doseči cilje znotraj načrtovanega časovnega okvirja projekta. Aktivnosti kontrole vključujejo merjenje opravljenega dela in za to potrebnega časa ter upravljanje z organizacijskim vidikom projekta, delegiranje nalog, spremembe časovnice, ažuriranje projektne dokumentacije ipd.

2.1 Opredelitev aktivnosti

Število aktivnosti in njihove vrste izhajajo iz managementa obsega projekta. Torej iz ciljev, ki so povezani z uporabno vrednostjo produkta projekta. Vhod v proces identifikacije aktivnosti je struktura WBS ter organizacijske in procesne lastnosti podjetja (PMI, 2004, str. 134). Celotno delo razdelimo na manjše obvladljive enote, ki same zase predstavljajo vsebinsko zaključeno celoto, katere stopnjo končanja je mogoče učinkovito meriti. Govorimo o opredelitvi aktivnosti.

Orodja in tehnike opredelitve aktivnosti so razčlenjevanje, uporaba vzorcev s preteklih projektov, ekspertna presoja ter postopno planiranje oz. planiranje v valovih, ki aktivnosti postopoma drobi na vse manjše in bolj podrobno, odvisno od tega, v kateri fazi je projekt.

Rezultat delitve je seznam aktivnosti v tabelarični obliki. Posamezna aktivnost ima več atributov, med katere uvrščamo identifikator aktivnosti, ime in kratek opis aktivnosti, morebitne predhodnike in naslednike, povezave do predhodnikov ter naslednikov, zahteve glede potrebnih virov, omejitve ter pomembne časovne atribute, kot so, npr. datum začetka ter konca aktivnosti ter njeno trajanje. Atributi aktivnosti so pomembni za aktivnosti managementa časa, ki sledijo opredelitvi (Schwalbe, 2010, str. 215). Izhod iz opredelitve je tudi seznam mejnikov projekta (angl. *milestone*), ki označujejo pomembne dogodke med izvajanjem projekta in so izbrani tako, da predstavljajo za deležnike projekta neko vrednost.

Členjenje aktivnosti je lahko različno podrobno in odvisno od velikosti projekta ter nivoja podrobnosti managementa projekta. Na nižjih nivojih in manjših projektih je s časovnega vidika dovolj členjenje na aktivnosti v trajanju enega do dveh delovnih tednov. Z vidika razumevanja vsebine aktivnosti, pa je potrebna bolj podrobna delitev. Podrobno členjenje aktivnosti pomaga pri ocenjevanju časa potrebnega za izvedbo, saj z dovolj podrobnim členjenjem pridemo do aktivnosti, ki jih poznamo s preteklih projektov in iz izkustev ocenimo njihovo trajanje.

2.2 Razvrščanje aktivnosti

Razvrščanje aktivnosti pomeni določanje vrstnega reda izvajanja ter zaključka posameznih aktivnosti in določanje vrste povezav med aktivnostmi. Za razvrščanje aktivnosti je potrebno razumeti vsebino vseh vključenih aktivnosti, poznati oz. imeti oceno trajanja vsake posamezne aktivnosti in poznati potrebne vire za končanje aktivnosti (Schwalbe, 2010, str. 217). Povezave oz. odvisnosti med posameznimi aktivnostmi so lahko:

- Obvezujoče, ko aktivnosti ne moremo opraviti pred končanjem nekaterih drugih. Npr. testiranja ne moremo opraviti pred končanjem objekta testiranja.
- Poljubne, ki izhajajo iz izkušenj in dobre prakse. Npr. podroben dizajn funkcionalnosti naprave ne bo začet pred potrditvijo specifikacij s strani stranke.

- Zunanje odvisnosti, ki izhajajo iz delovanja zunanjih subjektov, npr. dobaviteljev materiala, opreme ipd.

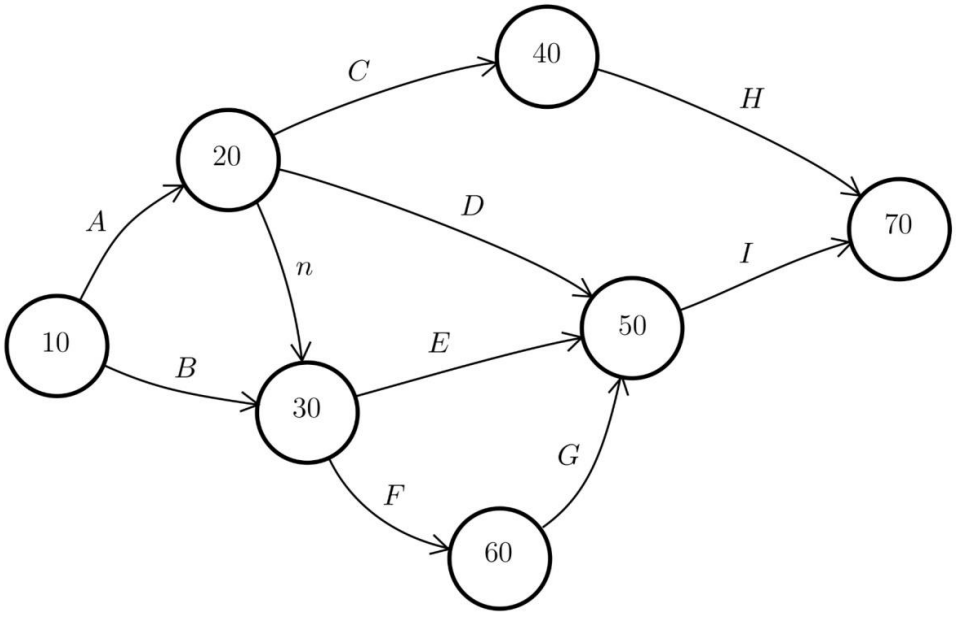
Razvrščanje aktivnosti je osnova za uporabo orodij za management časa, kot sta mrežni diagram aktivnosti in Gantt diagram.

Mrežni diagram aktivnosti

Mrežni diagram aktivnosti projekta (angl. *Activity Network Diagram* ali *Network Diagram*) grafično prikazuje povezave med aktivnostmi, ki jih je potrebno izvesti za uspešno končanje projekta in je orodje, ki je v široki uporabi med razvrščanjem aktivnosti. Ločimo 2 vrsti mrežnega diagrama:

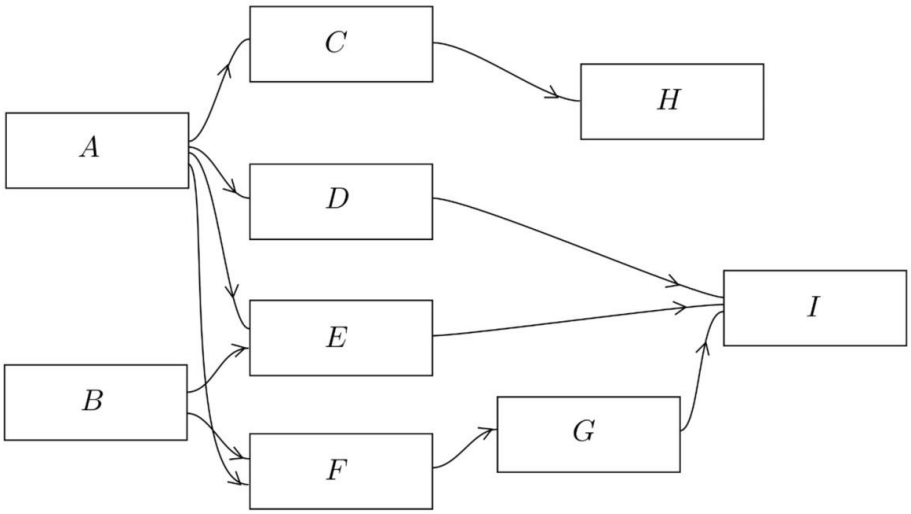
- AOA (angl. *Activity on Arrow*) diagram. Aktivnosti so v tej vrsti diagrama predstavljene s puščicami, ki povezujejo posamezna vozlišča. Vozlišča predstavljajo začetek in konec aktivnosti. Prvo vozlišče predstavlja začetek projekta in zadnje predstavlja njegov konec.
- PDM (angl. *Precedence Diagramming Method*) je pogosteje uporabljena metoda predstavitve poteka projekta. Posamezna aktivnost je v tej vrsti diagrama predstavljena s simbolom, kvadratom. Povezave, puščice med kvadrati določajo povezave oz. odvisnosti med aktivnostmi. Govorimo tudi o AON (angl. *Activity on Node*) diagramu.

Slika 3: AOA diagram projekta



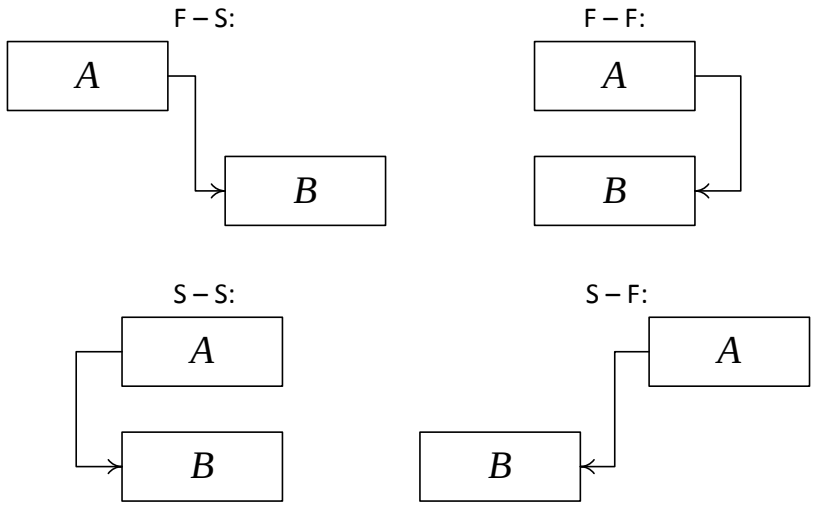
Vir: F. Solina, Projektno vodenje razvoja programske opreme, 1997, str. 76.

Slika 4: PDM diagram projekta



Vir: F. Solina, Projektno vodenje razvoja programske opreme, 1997, str. 77.

Slika 5: Tipi povezav med aktivnostmi projekta



Vir: Povzeto po K. Schwalbe, IT Project Management, 2010, str. 219.

V PDM diagramu se srečamo s povezavami med aktivnostmi, ki so lahko tipa:

- Konec-začetek (angl. *Finish-Start*) povezava, ki pomeni, da mora biti predhodna aktivnost zaključena, preden se lahko začne naslednja aktivnost.
- Začetek-začetek (angl. *Start-Start*) povezava, ki pomeni, da se aktivnost lahko začne šele, ko se začne z njo povezana aktivnost.
- Začetek-konec (angl. *Start-Finish*) povezava, ki pomeni, da je aktivnost lahko končana šele, ko se začne povezana aktivnost.
- Konec-konec (angl. *Finish-Finish*) povezava, ki pomeni, da je aktivnost lahko končana šele, ko je končana povezana aktivnost.

AON diagram oz. PDM metoda se uporablja pogosteje kot metoda AOA in ponuja pred slednjo vrsto prednosti (Schwalbe, 2010, str. 220):

- večina programskih orodij za vodenje projektov uporablja PDM metodo,
- PDM ne potrebuje praznih aktivnosti, ki jih AOA predstavi s črtkanimi puščicami,
- PDM metoda uporablja vse naštete povezave med aktivnostmi, medtem ko AOA metoda podpira le Konec-začetek povezavo.

Izhodi iz razvrščanja aktivnosti so mrežni diagram projekta v grafični obliki ter ažurirani projektni dokumenti, kot so seznam aktivnosti, njihovi atributi ter register tveganj.

2.3 Ocenjevanje potrebnih virov

Naloga ocenjevanja, za izvedbo aktivnosti potrebnih virov, je podrobna predstavitev vseh virov, ki so potrebni za vse predvidene aktivnosti projekta. Viri so materialni, človeški viri in oprema. Opredelitev potrebnih virov je narejena v tesni koordinaciji s planiranjem stroškov.

Vhodi v proces ocenjevanja so seznam aktivnosti, njihove lastnosti ter organizacijski vidiki podjetja, ki vplivajo na izbiro virov. Pomemben vhod je še t. i. koledar virov, ki opredeli, kdaj bodo kateri viri na voljo in kakšne so njihove lastnosti oz. usposobljenost v primeru človeških virov. Za človeške vire koledar opredeli delovne ter dela proste dni, planirane odsotnosti ipd. Za nečloveške vire koledar opredeli, v katerih obdobjih je možno koristiti neko opremo ali infrastrukturni objekt. Projektni manager s pomočjo koledarja virov ugotavlja, kdaj so določeni viri prosti ali zasedeni in na ta način sprejema odločitve povezane z načrtovanjem časovnice.

Za ocenjevanje potrebnih virov je pogosto v uporabi ekspertno ocenjevanje oz. svetovanje.

Ključno je predstaviti tipe potrebnih virov, njihovo število ter opredeliti, kdaj bodo predvidoma potrebni. Primerjava med potrebnimi viri ter viri, ki so na voljo, je pomemben vhod v ocenjevanje potrebnega časa ter predvsem v proces izdelave časovnice (Fuller et al., 2008, str. 253). Izhoda iz procesa ocenjevanja virov sta:

- seznam aktivnosti in potrebnih virov ter potrebna količina oz. njihovo število za vsako opredeljeno aktivnost ter
- seznam virov oz. struktura RBS (angl. *Resource Breakdown Structure*), ki je struktura, ki predstavlja vse potrebne vire. Predstavi tipe virov ter v kolikšnem številu bodo potrebni. Nekateri viri bodo namreč potrebni v celotnem trajanju projekta, medtem ko bodo drugi potrebni le v nekaterih fazah.

Izhodi iz procesa so še ažurirani projektni dokumenti, kot je npr. seznam aktivnosti in njihovi atributi ter koledar virov.

Na projektih, ki vključujejo veliko število ter različne vrste človeških virov in kjer velik del stroškov predstavljajo stroški dela, je pomembno preučiti vse možne alternative, kako človeške vire tem učinkoviteje izkoristiti. Tudi z viri povezanim vidikom projekta se je potrebno posvetiti v zgodnji fazi projekta (Schwalbe, 2010, str. 222).

2.4 Ocenjevanje trajanja aktivnosti

Ocenjevanje trajanja aktivnosti je proces, ki oceni koliko časa je potrebno za končanje vsake posamezne aktivnosti, pri znanem obsegu dela, znanih potrebnih virih za končanje aktivnosti ter znani informaciji o tem, kdaj so le-ti na voljo. Poleg naštetega sta vhod v proces ocenjevanja še lastnosti podjetja ter izkušnje s preteklih projektov.

Na področju informacijske tehnologije obstaja več razvrstitev ocenjevalnih tehnik, ki tehnike delijo v skupine. Med bolj znane delitve štejemo razvrstitev po ISBSG (angl. *International Software Benchmarking Standards Group*), razvrstitev Boehm, razvrstitev Myrteveit, Jørgensenovo razvrstitev ter nekatere druge. Jørgensen deli tehnike ocenjevanja na ocenjevanje s pomočjo ekspertne presoje, ocenjevanje s pomočjo formalnih modelov ter kombinirano ocenjevanje. Večina projektov s področja razvoja programske opreme je ocenjena s pomočjo ekspertne presoje (Živković, 2011, str. 28).

Ekspertno ocenjevanje zajema analogno metodo, delitev celotnega dela na manjše enote dela, ocenjevanje od spodaj navzgor ali od zgoraj navzdol, tehniko Delfi ter druge. Ocenjevanje temelji na intuiciji, kar je glavna razlika v primerjavi z ocenjevanjem po formalnih metodah.

Pri ocenjevanju potrebnega časa po analogni metodi, se projektni manager poslužuje uporabe znanih ocen podobnih aktivnosti na preteklih projektih ter primerja tedanjo oceno potrebnega časa z dejansko porabljenim časom za aktivnost. Iz primerjave ugotovi tendenco ocenjevanja, ali je ta bila oz. je optimistična ali pesimistična. V primeru ugotovljenih odstopanj iz preteklosti se oceno aktivnosti na trenutnem projektu ustrezno prilagodi.

Formalne metode ocenjevanja trajanja aktivnosti temeljijo na avtomatizaciji pridobivanja ocene. Formalno ocenjevanje je manj uporabljeno, delimo pa ga v skupine, ki so ocenjevanje temelječe na parametričnem modelu, ocenjevanje temelječe na modelu strojnega učenja ter kombinirano ocenjevanje. Formalno ocenjevanje običajno izvajamo v dveh stopnjah. V prvi stopnji na ocenjevanju velikosti projekta ter v drugi stopnji na ocenjevanju obsega projekta z upoštevanjem dejavnikov, kot je, npr. produktivnost izvajalske ekipe.

Ocenjevanje potrebnega časa za izvedbo aktivnosti ne gre zamenjevati s trajanjem aktivnosti. Trud oz. potreben napor (angl. *effort*) pomeni delo, ki ga je potrebno vložiti za izvedbo aktivnosti. Trajanje (angl. *duration)* opredeli skupen čas od začetka dela na aktivnosti do njenega zaključka.

Izhoda iz procesa ocenjevanja aktivnosti sta seznam aktivnosti z ocenjenim vložkom dela za končanje aktivnosti in trajanjem aktivnosti ter ažurirani projektni dokumenti.

2.5 Izdelava časovnice projekta

Priprava časovnice temelji na izvajanju razvrščanja aktivnosti, ocenjevanju trajanja aktivnosti, ocenjevanju potrebnih virov ter na informacijah o začetku in predvidenem koncu projekta. Priprava časovnice, ki je ustrezna projektu, je končana po več iteracijah. Cilj je pripraviti časovnico, ki bo temeljila na realnih ocenah in bo imela možnost biti izvedena v zastavljenih časovnih omejitvah. S časovnico mora biti možno zastavljene cilje in napredek na projektu učinkovito spremljati ter izvajati kontrolo nad izvedbo s pravočasnim odzivom na nastale težave.

Vhodi v izgradnjo časovnice so seznam ter atributi aktivnosti, ocenjeno trajanje aktivnosti, mrežni diagram aktivnosti, seznam in koledar potrebnih virov ter lastnosti organizacije, ki vplivajo na projekt.

Orodja in metode, ki se uporabljajo pri pripravi časovnice so (Schwalbe, 2010, str. 153):

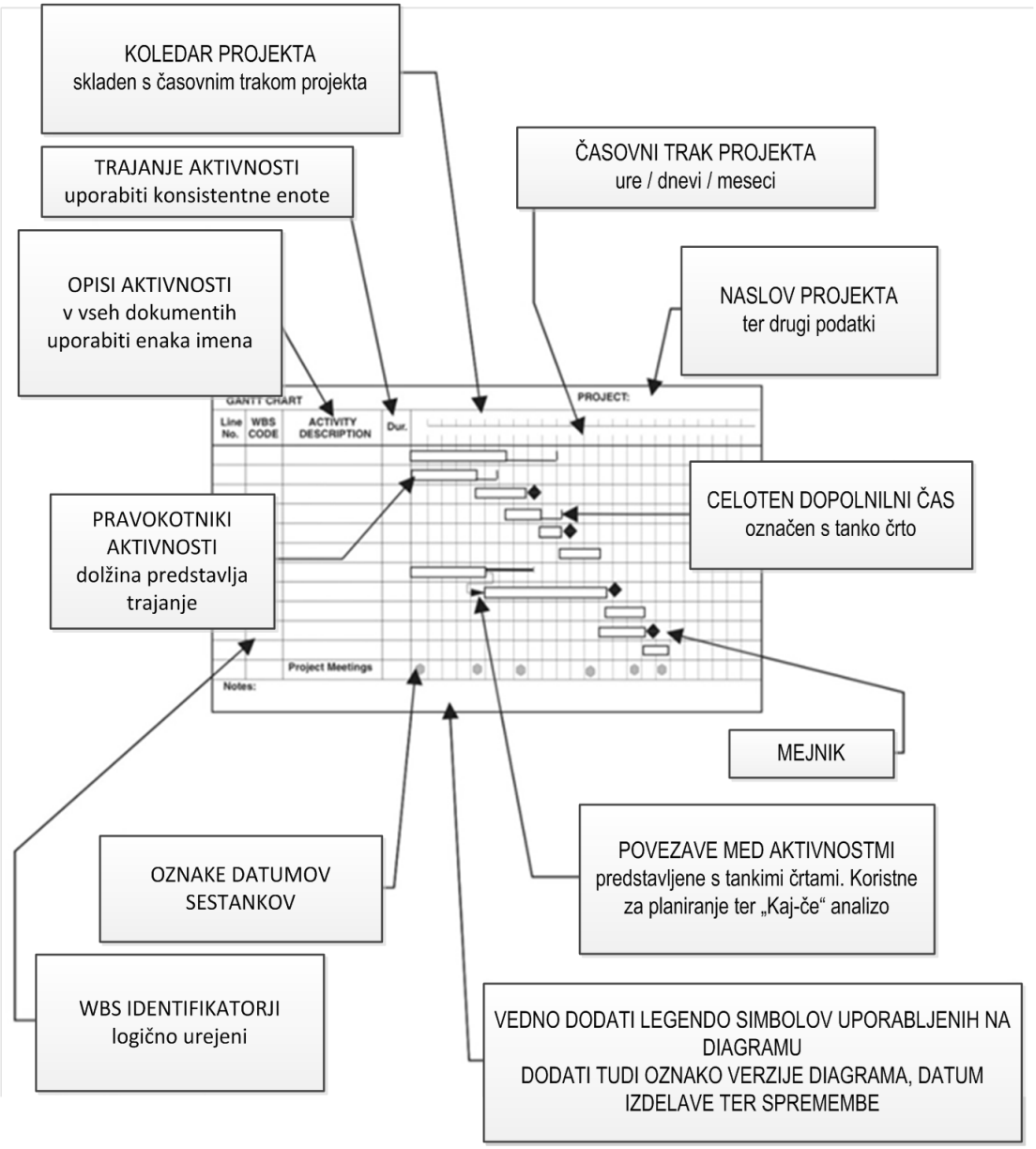
- časovna analiza mreže,
- metoda kritične poti (angl. *Critical Path Method*, v nadaljevanju CPM),
- metoda kritične verige (angl. *Critical Chain Scheduling*, v nadaljevanju CSS),
- kaj-če analiza (angl. *What-if Analisys*),
- tehnike krajšanja časovnice, kot sta časovno prekrivanje (angl. *Fast Tracking*) ter skrajno stiskanje trajanja (angl. *Crashing*), ki so uporabljane tudi med kontrolo poteka projekta

in še (PMI, 2010, str. 145):

- izravnavanje obremenitve virov (angl. *Resource Leveling*),
- programska oprema za projektno vodenje,
- uporaba koledarjev,
- prilagajanje prehitkov in zakasnitev ter
- model terminskega plana.

Izhod iz procesa izdelave časovnice je časovnica, ki je predstavljena grafično v obliki t. i. Gantt diagrama. Časovnica projekta oz. diagram je običajno izdelan z računalniškimi orodji namenjenimi managementu projektov. Diagram vsebuje informacije o mejnikih projekta, prikazuje začetek in konec aktivnosti ter njihove medsebojne povezave. Diagram je lahko glede na potrebe projekta bolj ali manj podroben.

Slika 6: Primer Gantt diagrama s pomeni elementov



Vir: T. L. Young, The Handbook of Project Management, 2007, str. 153.

Uporaba Gantt diagrama je priporočljiva iz razlogov (Silva, 2012):

- Diagram učinkovito vizualno prikaže potek projekta ter preprečuje zmedo, katere aktivnosti so bile končane in katere ne, ter nazorno prikaže tudi stopnjo dokončanja vsake posamezne aktivnosti.
- Informacija o poteku projekta je na voljo na enotnem diagramu in hkrati vsem deležnikom.
- Diagram nazorno prikaže povezave med aktivnostmi ter učinkovito identificira, katere aktivnosti je potrebno končati za napredovanje projekta.
- Diagram nazorno kaže, kateri viri so v uporabi ter kateri viri bodo potrebni za naslednje aktivnosti ter omogoča pravočasno rezervacijo virov.

- Diagram omogoča učinkovit pregled prihodnjih aktivnosti in faz projekta ter omogoča tistim, ki sprejemajo odločitve pravočasno pripravo naslednjih korakov.

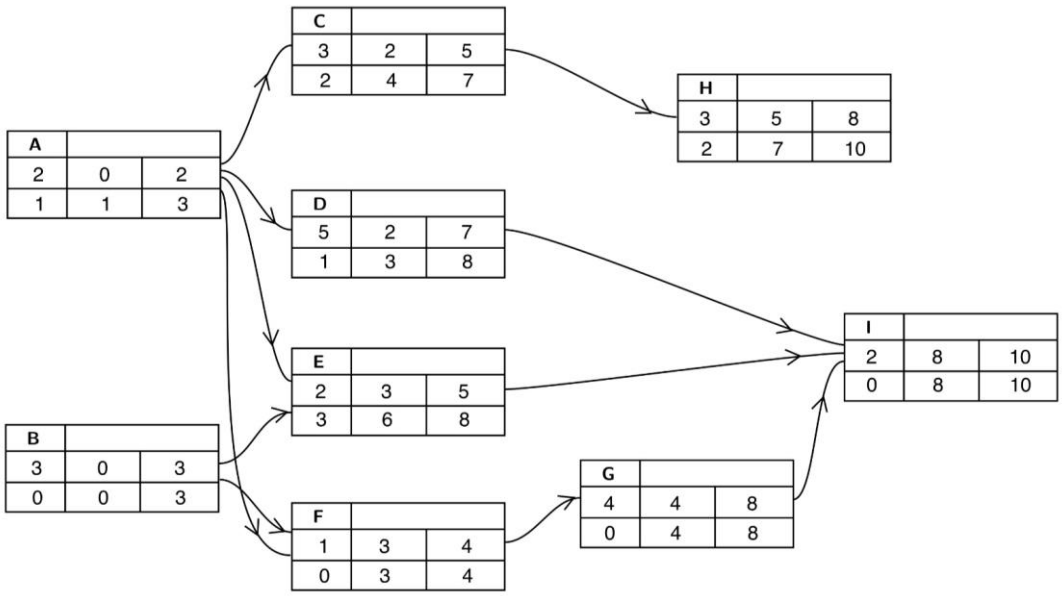
Drugi izhodi procesa so osnovna časovnica (angl. *base schedule*), potrjena s strani deležnikov projekta, ki vsebuje le nekaj datumov, kot sta začetek in konec projekta, časovnica, ki kaže zasedenost virov, alternativne časovnice, ažurirani projektni dokumenti ter register tveganj.

Metoda kritične poti

Metoda kritične poti oz. CPM je metoda, s katero ocenjujemo celotno trajanje projektov in preprečujemo časovne prekoračitve projektov. Projekt ima navadno več paralelnih poti in aktivnosti, ki se izvajajo vzporedno. Z metodo v dveh korakih, z računanjem naprej ter računanjem nazaj, izračunamo teoretično najzgodnejše ter najkasnejše začetne in končne datume aktivnosti za vse možne poti. Omejenosti virov ne upoštevamo. Schwalbejeva (2010, str. 228) opredeli kritično pot projekta kot zaporedje aktivnosti projekta, ki določa najkrajši čas, v katerem je lahko projekt končan. Je hkrati najdaljša pot skozi mrežni diagram projekta in nima pomičnosti (angl. *float* ali *slack*). Aktivnosti na tej poti imenujemo kritične aktivnosti.

Pomičnost je čas, za katerega lahko konec aktivnosti zamudi, da ta zamuda ne vpliva na datum končanja projekta. Aktivnosti, ki so na kritični poti, bodo v primeru, da bodo končane pozno, vplivale na končni datum končanja projekta. Pot, ki vključuje kritične aktivnosti, je gonilo projekta, vendar projekt ni končan, dokler niso končane vse aktivnosti projekta (Schwalbe, 2010, str. 228).

Slika 7: Kritična pot projekta



Vir: F. Solina, Projektno vodenje razvoja programske opreme, 1997, str. 83.

Slika 7 prikazuje kritično pot projekta, ki jo sestavlja zaporedje aktivnosti B-F-G-I. Kritična pot na sliki nima pomičnosti, ki je označena v pravokotnikih aktivnosti spodaj levo. Drugi elementi, v zgornji vrstici pravokotnikov aktivnosti, so njeno trajanje, zgodnji začetek in zgodnji konec. V spodnji vrstici pa pomičnost, pozen začetek ter pozen konec aktivnosti.

Pred kalkulacijo kritične poti je potrebno narediti kakovosten mrežni diagram, ki temelji na temeljiti pripravi osnovani na WBS strukturi. Potrebno je oceniti trajanje vsake posamezne aktivnosti ter nato izračunati kritično pot projekta.

Ključno je, da se projektni manager zaveda te poti in brani aktivnosti na njej, npr. s prerazporejanjem virov na aktivnosti na kritični poti. Da ne pride do zamude izvedbe projekta oz. njegovega zaključka, je aktivnosti na kritični poti projekta potrebno ščititi z ukrepi, kot so časovno prekrivanje ter skrajno stiskanje trajanja časovnice in drugimi.

Zaradi ščitenja kritičnih aktivnosti, se je pomembno zavedati, da se kritična pot med izvajanjem projekta lahko spremeni, ko projekt ne teče po načrtu. Zato je potrebno med samim projektom večkrat na novo preračunati kritično pot ter identificirati morebitne nove kritične aktivnosti, jih ščititi in obraniti projekt pred zamudo.

Metoda kritične verige

Metoda kritične verige (angl. *Critical Chain Scheduling* v nadaljevanju CSS) je tehnika s katero projektni managerji izpolnijo dva od treh glavnih ciljev oz. izzivov s katerimi se srečujejo v projektnem managementu. To sta čim hitrejše končanje projekta ter izvedba čim večjega števila projektov brez dodatnih virov (Kerzner, 2003, str. 534). Tehnika velja za kompleksno, vendar močno ter eno pomembnejših orodij, ki so bila razvita v zadnjem obdobju.

Tehnika temelji na teoriji omejitev (angl. *Theory of Constraints*), ki jo je razvil Eliyahu M. Goldratt. Teorija pravi, da ima vsak sistem svoj najšibkejši člen in da je za uspeh potrebno ta člen identificirati, ga ves čas nadzorovati ter izboljšati.

Prvi pomemben princip kritične verige je omejenost virov. Za projektni management to pomeni, da se med izdelavo časovnice zavedamo omejenosti virov ter najdemo dodaten vir ali pa aktivnosti izvajamo zaporedoma. Tipičen primer omejenosti virov je kos opreme, ki ga potrebujemo za izvedbo dveh aktivnosti. Dva druga principa sta uporaba vzporednega izvajanja nalog ter drugačna uporaba časovnih blažilnikov (angl. *buffers*).

Tehnika CCS ne predvideva uporabe hkratnega delovanja določenega vira na vzporednih aktivnostih (angl. *multitasking*) oz. teži k čim manjši uporabi takšnega načina dela. Vir naj ne bi bil delegiran na dve ali več vzporednih aktivnosti, ker bodo posledično zamujale vse oz. bo njihovo trajanje daljše od trajanja vloženega dela. V primeru uporabe hkratnega delovanja morajo aktivnosti imeti jasno določene prioritete ter znano informacijo, katere

aktivnosti so višje prioritete od drugih. Neuporaba takšnega načina dela ima pozitiven učinek, saj se med izvajanjem izognemo konfliktom, ki sledijo iz rezervacij virov in ne tratimo časa, ki je potreben za premik pozornosti iz ene aktivnosti na drugo.

CCS predvideva uporabo časovnih blažilnikov na drugačen način, kot je običajno. Namesto časovnih rezerv, ki so del ocene vsake aktivnosti predvideva uporabo projektne časovne rezerve na koncu projekta ter uporabo rezervnega časa, ki je umeščen pred tiste aktivnosti, ki so identificirane kot kritične, vendar njihovi predhodniki niso kritične aktivnosti (angl. *feeding buffer*). S tem, da aktivnosti nimajo lastnih rezerv, se projektni manager izogne učinku t. i. Parkinsonovega zakona, ki pravi, da se izvedba aktivnosti vedno zavleče v celoten predviden čas izvedbe. Uporaba drugačnega principa časovnih blažilnikov ščiti pravo časovno omejitev, ki je rok zaključka projekta (Schwalbe, 2010, str. 233).

PERT tehnika

Ko je nedoločenost trajanja posameznih aktivnosti projekta velika, se v namen zmanjšanja nedoločljivosti pri ocenjevanju trajanja aktivnosti, poslužujemo enostavne, vendar učinkovite 3-točkovne tehnike ocenjevanja, PERT. Tehnika vključuje uporabo optimistične, pesimistične ter najverjetnejše ocene potrebnega časa. Z zmanjšanjem nedoločljivosti omejimo tveganje, ki nastane zaradi nepravilno ocenjenega trajanja aktivnosti in bi negativno vplivalo na pravilno pripravo časovnice ter izvedbo projekta.

Ocenjevanje potrebnega časa izvedemo po enačbi (Solina, 1997, str. 70):

$$t_e = \frac{t_o + 6 \times t_{ml} + t_p}{6} \qquad (1)$$

kjer so:

t_e ocenjeno trajanje (angl. *estimated*)

t_o optimistična napoved trajanja (angl. *optimistic*)

t_{ml} najverjetnejša napoved trajanja (angl. *most likely*)

t_p pesimistična napoved trajanja (angl. *pesimistic*)

Varianca ocenjenega trajanja aktivnosti je:

$$v = (t_o - t_e)^2 + 4 \times (t_{ml} - t_e)^2 + (t_p - t_e)^2 \qquad (2)$$

PERT uvede uteženo povprečje, saj so optimistična, pesimistična ter najverjetnejša ocena potrebnega časa za končanje posamezne aktivnosti utežene z različnimi utežmi. Zmanjša tveganje zaradi nepravilne ocene posamezne aktivnosti in je v pomoč pri pripravi najbolj realistične časovnice projekta.

Kaj-če analiza

Kaj-če analiza je statistična metoda, ki se uporablja za določanje vpliva spremembe vhodnih podatkov na izhode sistema. Sistem je predstavljen z modelom, ki določa odvisnosti med vhodi in izhodi sistema.

Za uporabo Kaj-če analize je potrebno končati več faz. Potrebna je preslikava realnega problema v model in razvoj ustreznega modela. Nato je potrebno zagotoviti ustrezno količino kakovostnih podatkov. Zbiri vhodnih podatkov predstavljajo scenarije, ki jih želimo simulirati.

Izvedba simulacij da rezultat analize, ki je napoved kako se bo sistem odzval na različne vhodne parametre.

2.6 Kontrola poteka projekta

Kontrola časovnice

Kontrola časovnice pomeni, na osnovi merljivih dejstev, poznati trenuten status projekta, nadzorovati faktorje, ki vplivajo na njegovo napredovanje ter izvajati management sprememb, ko se le-te zgodijo ali so nujno potrebne. Najpomembnejši vhodi v uspešno kontrolo časovnice so projektni načrt, sama časovnica projekta ter poročila o učinkovitosti dela na aktivnostih.

Orodja, ki se najpogosteje uporabljajo za kontrolo časovnice, so:

- poročila o napredovanju aktivnosti,
- računalniška orodja za management projektov,
- primerjave med planiranim in dejanskim stanjem opravljenih del z uporabo orodij, kot je Gantt diagram za časovno primerjanje (angl. *Tracking Gantt Chart*),
- prerazporejanje virov oz. izravnava obremenitve virov (angl. *Resource Leveling*) ter splošno upravljanje s človeškimi viri,
- kaj-če analiza,
- tehnike krajšanja časovnice, kot sta časovno prekrivanje ter skrajno stiskanje trajanja,
- metode za merjenje napredka projekta, kot je, npr. metoda prislužene vrednosti (angl. *Earned Value Management,* v nadaljevanju EVM) ter druge.

Izhodi iz procesa kontrole časovnice so ocene o opravljenem delu, ukrepi, ki so del organizacijskega vidika projekta, zahteve po spremembah managementa projekta ter popravki in uskladitve projektne dokumentacije (Schwalbe, 2010, str. 237).

Projektni manager preko kontrole časovnice in odstopanja od načrtovanega napredka identificira prisotnost težav in potrebo po korektivnih ukrepih. Analiza stanja identificira

vzroke za odstopanje, ki jih je potrebno z ukrepi odpraviti ter hkrati najti rešitev za obstoječe stanje. Korektivni ukrepi so lahko (Young, 2007, str. 212):

- prerazporeditev izvajanja aktivnosti na druge manj zasedene vire, ki razbremenijo preobremenjene vire in pomagajo k lovljenju mejnikov,
- poskus motivirati izvajalce k opravljanju dodatnega dela,
- prerazporeditev dodatnih virov na kritične aktivnosti,
- premaknitev mejnikov in rokov za končanje projekta z možnostjo nadomestitve zamud kasneje v projektu,
- zmanjšanje obsega in/ali kakovosti predvidenih rezultatov projekta.

Premaknitev mejnikov, zmanjšati obseg projekta ter nekateri drugi ukrepi so možni le z dovoljenjem drugih deležnikov projekta in jih je težje izvesti, ko zadevajo aktivnosti na kritični poti. Pomembna je tudi širša evalvacija ukrepov oz. kako posamezen ukrep vpliva na poslovni načrt projekta. Premaknitev mejnikov običajno pomeni neuspeh in predčasen zaključek projekta.

Ukrepi so evalvirani in sprejeti v naštetem vrstnem redu. Vplivajo na stroške projekta in zato je hkrati z njihovim vpeljevanjem, nujen tudi nadzor stroškov ter odobritev sponzorja projekta, da so stroški ukrepov upravičeni.

Metode za krajšanje časovnice

Namen v nadaljevanju opisanih metod je rešitev težav, ki so povezane s trenutno prekoračitvijo porabljenega časa na projektu in zahtevajo ustrezne ukrepe. Na projektu se največ časa posveča reševanju težav, ki so povezana s kontrolo časa (Young, 2007, str. 214). Delo poteka več časa, kot je bilo planirano, bodisi zaradi nepravilne ocene ali zaradi nepredvidenih dogodkov. Metode, ki so največkrat v uporabi, so časovno prekrivanje, skrajno stiskanje trajanja ter hkratno izvajanje nalog.

Skrajno stiskanje trajanja je tehnika, s katero skrajšamo časovnico projekta z najnižjimi dodatnimi stroški. Primer je sorazmerno povečanje obsega dela, v obliki izvedenih ur, na aktivnosti, ki bi v originalni časovnici trajala dlje. Aktivnost je ob dodatnem delu končana v krajšem času, stroški projekta pa se ne povečajo bistveno, saj celotna količina dela ostane enaka. Pomemben je tudi ponoven izračun kritične poti, ki lahko da nov datum končanja projekta (Schwalbe, 2010, str. 233).

Časovno prekrivanje je tehnika, s katero projekt, ki je v zaostanku, pospešimo. Časovno prekrivanje pomeni, da aktivnosti, ki se po originalnem načrtu izvajajo v zaporedju, izvajamo paralelno. Tehnika hkrati poveča stopnjo tveganja za nastanek težav, ki se manifestirajo v nadaljnji zamudi ali tako, da je potrebno ponovno delo (angl. *rework*). Tveganje se poveča, ker v trenutku začetka dela pogostokrat nimamo na voljo vseh vhodov v vzporedno izvajani aktivnosti. Podjetja, ki so projektno organizirana, pogosto uporabljajo

tehniko, vendar je narobe, če tehnika postane privzet način dela v izvajanju projektov (Kerzner, 2003, str. 475).

Hkratno izvajanje nalog je okoliščina, ko je vir delegiran na vsaj 2 aktivnosti hkrati. Situacija je običajno nezaželena, vendar običajna na področju projektnega managementa. Vir je lahko delegiran na dve aktivnosti hkrati na istem projektu ali pa je v podjetju, v katerem poteka več vzporednih projektov, razporejen na vsaj dve aktivnosti na dveh različnih projektih. Primer je človeški vir, ki ima specifična znanja potrebna več aktivnostim ali pa je trenutni projekt v fazi zaključevanja, management podjetja pa si želi začeti z novim projektom, ki bo deloma tekel vzporedno prvotnemu.

Metoda prislužene vrednosti

Nadzor nad izvajanjem celotnega projekta zahteva od projektnega managerja imeti nadzor nad stroški, izvedenim delom ter časom. Metoda prislužene vrednosti EVM, se je v preteklosti izkazala za učinkovito pomoč managerjem pri razumevanju, v kakšnem stanju je projekt oz. kako le-ta napreduje.

EVM temelji na izračunu vrednosti, ki je bila med izvajanjem projekta ustvarjena iz uporabljenih virov, tj. denarja, vloženega dela ter časa, in primerjave s planirano vrednostjo za trenutek, ko je izračun narejen. Za izračun kazalcev stanja projekta je potrebno poznati planirane stroške ter vrednost planiranih del projekta. Kazalci stanja projekta so enostavno izračunani kot deleži med vrednostmi in kažejo, kako uspešno projekt napreduje, ali je pred planom ali za planiranim zaostaja.

Ločimo kazalce, ki so v neposredni povezavi s stroški projekta, kazalce, ki so v povezavi s časovnico projekta oz. terminskim planom, ter kazalce v povezavi z vloženim delom. Kazalci v povezavi s stroški projekta so:

- dejanski stroški AC (angl. *Actual Cost*), ki pomeni dejanske stroške projekta, ki jih je povzročilo izvajanje del.
- planirani stroški BAC (angl. *Budget at Completion*), ki pomeni celotne planirane stroške ob zaključku projekta. BAC je vsota vseh PV.
- prislužena vrednost EV (angl. *Earned Value*), ki pomeni planirani znesek stroškov za že opravljeno delo. Vrednost je vsota vseh končanih ali delno končanih del. Vrednost posameznega dela je njegova ocenjena vrednost ob planiranju.
- planirana vrednost PV (angl. *Planned Value*), ki pomeni planirani strošek za planirano delo, ki ga je potrebno opraviti za aktivnost.
- stroškovni indeks CPI (angl. *Cost Performance Index*), ki pomeni razmerje med planirano vrednostjo prisluženih del (EV) ter dejansko prisluženo vrednostjo del (AC).
- stroškovni odmik CV (angl. *Cost Variance*), ki pomeni razliko vrednosti med planiranimi stroški in stroški, ki so dejansko nastali med izvajanjem del.

- napoved EAC (angl. *Estimate at Completion*), ki pomeni strošek projekta ob njegovem zaključku ob ohranjanju trenutne učinkovitosti.

Pomen kazalcev je naslednji:

- Če je CPI enak 1, pomeni, da je projekt na planirani poti, CPI večji od ena pomeni biti pred planom in CPI manjši od 1 pomeni biti v zaostanku.
- Če je CV enak 0 pomeni, da je projekt na planirani poti, CV večji od 0 pomeni, da je projekt cenejši od planiranega ter CV manjši od 0 pomeni, da je projekt dražji in je potrebno zagotoviti dodatna sredstva, zmanjšati obseg projekta ipd.
- Če je EAC enak BAC pomeni, da so stroški projekta skladni s planiranimi. EAC manjši od BAC pomeni, da je projekt cenejši kot je bilo planirano (CPI > 1) ter če je EAC večji od BAC pomeni, da je projekt dražji kot je bilo planirano (CPI < 1).

V povezavi s časovnico projekta sta kazalca:

- terminski indeks SPI (angl. *Schedule Performance Index*), ki je razmerje med časom izvajanja projekta ter planiranim časom po časovnici projekta. Kazalec pomeni učinkovitost izvajanja projekta s časovnega vidika.
- odmik terminskega plana SV (angl. *Schedule Variance*), ki pomeni razliko vrednosti prisluženih in planiranih del glede na časovnico projekta.

Pomen kazalcev je naslednji:

- Če je SPI enak 1, pomeni, da je projekt na planirani poti, SPI večji od ena pomeni, da je izvajanje projekta nad pričakovanji ter SPI manjši od 1 pomeni izvajanje projekta pod pričakovanji.
- Če je SV enak 0 pomeni, da je projekt na planirani poti, SV večji od 0 pomeni, da je izvajanje projekta nad pričakovanji ter SV manjši od 0 pomeni, da je izvajanje projekta pod pričakovanji in je zakasnitev zaključka projekta verjetna.

V povezavi s stroški ter vložkom dela sta kazalca:

- TCPIB (angl. *To Complete Performance Index within Budget*), ki izračuna potrebno učinkovitost, da bi bil projekt končan znotraj predvidenih stroškov ter
- TCPIP (angl. *To Complete Performance Index within Projected Estimate to Complete*), ki predvidi izid projekta ob konstantni oz. nespremenjeni učinkovitosti.

Čeprav je metoda EVM v uporabi že prek 50 let, ji je očitano, da so tudi kazalci, ki so v povezavi s časovnico, osnovani na stroških in ne na času. Iz tega razloga Lipke in Henderson trdita, da metoda ni primerna za projekte, ki trajajo preko planiranih časovnih okvirjev (v Peters, 2008, str. 213). Vpeljana je bila razširitev metode, označena s kratico ES (angl. *Earned Schedule*). ES je čas, v katerem bi, v skladu z načrtom, morala biti prislužena vrednost del EV. ES se določi tako, da ob trenutnem času trajanja projekta,

naredimo projekcijo vrednosti prislužene vrednosti del na krivuljo planirane vrednosti. Način projekcije prikazuje slika 8. Navpična projekcija presečišča na časovno os pomeni dejansko prisluženo časovnico ES.

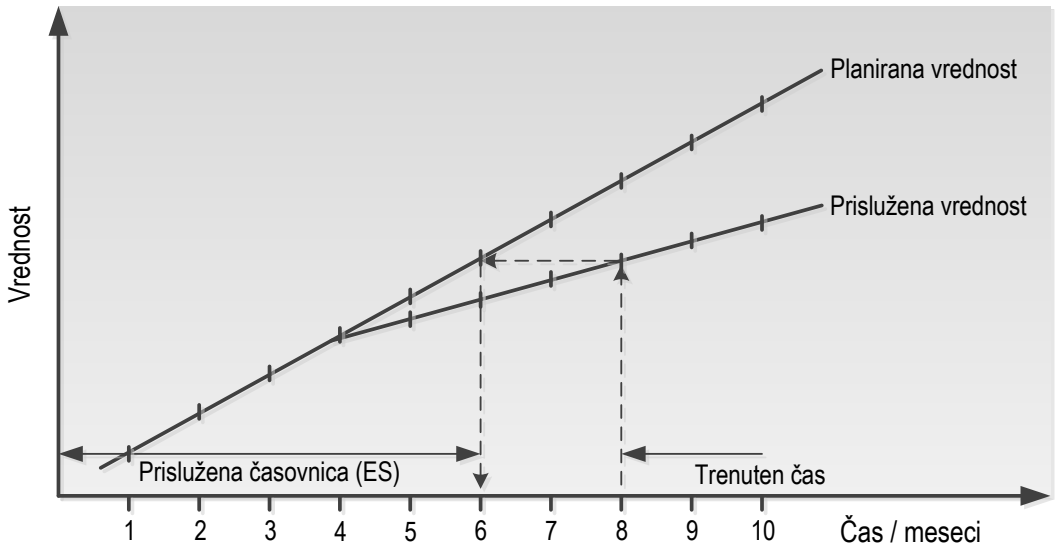
Pomembna lastnost EVM metode in njene razširitve ES je v tem, da povezuje delo, čas ter stroške na način, da sprememba enega parametra vpliva na druge. Da je uporaba metode učinkovita sta potrebni (Peters, 2008, str. 214; Fuller et al., 2008, str. 200):

- dovolj podrobna delitev del na enote na osnovi strukture WBS, ki jim je enostavno določiti status ali so končane ali ne oz. v kolikšni meri so končane ter
- realno in objektivno določanje ocen.

ES razširitev prinese 2 koristi (Lipke, 2012, str. 9):

- odstrani težavno potrebo po interpretaciji časa skozi stroške ter
- odstrani pomanjkljivost SV ter SPI kazalcev metode za končan projekt, ki je zamudil zaključek. V tem primeru da EVM metoda neustrezne rezultate. Kazalec SV je enak 0 in kazalec SPI je enak 1, kar predstavlja popolno izvedbo z vidika časa, kljub temu, da je projekt imel zamudo.

Slika 8: Prikaz delovanja metode prislužene časovnice



Vir: Povzeto po L. J. Peters, Getting Results From Software Development Teams, 2008, str. 213.

Pozitivna lastnost razširitve ES je tudi, da za uporabo ne potrebuje nobenih dodatnih informacij v primerjavi z EVM metodo.

Ključna je pravočasna in stalna uporaba metode, ki bo dovolj zgodaj identificirala zamude pri izvajanju projekta in bo omogočila pravočasno ukrepanje, da bi pripeljali projekt nazaj na planirano pot.

3 MANAGEMENT KAKOVOSTI

V zadnjih letih je prišlo do izjemne spremembe v pojmovanju kakovosti, kot tudi do sprememb procesov, ki posledično vodijo do vse kakovostnejših produktov. V spremembe v dojemanju kakovosti in njenega dviga so podjetja prisiljena v obdobju ekonomskih kriz, ko ni mogoče prodati vseh ponujenih produktov. V času kriz odjemalci dvignejo svoja pričakovanja glede produktov, postanejo bolj preudarni, zahtevajo produkte, ki jim bodo zadovoljili več potreb in prihranili stroške. Daljše ekonomske krize lahko preživijo le podjetja, ki se nanje ustrezno odzovejo in postanejo učinkovitejša. Učinkovitost lahko dosežejo preko dviga kakovosti svojih produktov, ki jim na daljši rok pomaga nižati stroške poslovanja. Kakovost namreč, glede na številne avtorje s področja kakovosti, podjetje stane manj kot kasnejše odpravljanje napak zaradi nekakovosti ali izgube ugleda. Prvi avtor, ki je takšno idejo zagovarjal je bil Philip B. Crosby. Tezo je razvil v trditev, da kakovost ne stane nič (Fuller et al., 2008, str. 286). Proces izboljšave kakovosti produktov temelji na:

- Planiranju za kakovost, ki pomeni, da podjetje identificira svoje odjemalce ter njihove potrebe ali identificira svoje produkte, s katerimi lahko tem potrebam kar najbolje zadosti.
- Organiziranju svojih struktur, tako notranjih kot tudi zunanjih. V ospredju je predvsem organizacijska struktura podjetja in njeni delovni procesi.
- Spremljanju kakovosti ter pravočasnem odzivanju na situacije, ko produkt ne zadovolji pričakovanj strank ali objektivnih meril kakovosti.

Management kakovosti, kot področje projektnega managementa, sestavljajo trije glavni procesi (PMI, 2004, str. 189):

- Planiranje kakovosti, ki pomeni identifikacijo potrebnih norm in standardov, ključnih za kakovost produkta. Pomeni tudi predvidevanje situacij med izvajanjem projekta, ki bodo vplivale na kakovost produkta in pripravo ustreznih odzivov nanje, da bi bili izhodi iz aktivnosti v skladu s pričakovanimi. V praksi to pomeni izbiro ustreznih postopkov, orodij, materialov, vpeljavo standardov in metrik kakovosti v procese, zagotavljanje usposabljanja izvajalcev ter vzpodbujanje zavedanja o pomenu kakovosti vseh udeležencev projekta. Kakovost je potrebno v izdelek vgraditi in planiranje kakovosti je prvi korak k temu cilju.
- Izvajanje aktivnosti zagotavljanja kakovosti, kar pomeni imeti sistematičen pristop usmerjen k temu, da bo produkt ponovljivo izdelan v skladu z zahtevami. Izvajanje zagotavljanja kakovosti zahteva aktivno udeležbo odgovornih oseb skozi celoten življenjski cikel projekta. Največji delež odgovornosti pri zagotavljanju kakovosti mora prevzeti vrhnji management podjetja, hkrati pa je za dosego cilja potrebna aktivna udeležba vseh zaposlenih.
- Izvajanje aktivnosti kontrole kakovosti, ki pomeni primerjanje rezultatov s pričakovanimi in prilagajanje obstoječih procesov s ciljem doseči zastavljene

kakovostne cilje. Proces povezujemo s tehničnimi postopki in orodji za kontrolo kakovosti in uporabo metrik določenih v fazi planiranja.

3.1 Opredelitev in pristopi k definiciji kakovosti

Podjetja ter odjemalci priznavajo, da je kakovost težko univerzalno opredeliti. V literaturi najdemo 4 teoretične managerske pristope h kakovosti. Skupna lastnost teh pristopov je, da se kakovost smatra kot proces, ki ga je mogoče planirati, upravljati ter kontrolirati z uporabo managerskih in tehničnih znanj. Meje med posameznimi pristopi so večkrat navidezne in se v mnogočem prekrivajo (Kelemen, 2003, str. 9). Ločimo:

- Pristop z vidika izdelka, ki pojmuje kakovost kot merljivo lastnost izdelka. Posamezne lastnosti izdelka je mogoče meriti in tako meriti njegovo skupno kakovost ter oceniti kolikšno bo zadovoljstvo pri uporabi izdelka.
- Pristop z vidika izdelave, ki pojmuje kakovost kot stopnjo, s katero posamezen izdelek ustreza zahtevam ali specifikacijam. Cilj pristopa je s statističnimi metodami ter inšpekcijo izdelkov narediti proces izdelave stabilen in ponovljiv ter posledično priti do dviga kakovosti. Pristop vpelje v proces izdelave pojem kontrole kakovosti ter pojem standardizacije.
- Pristop z vidika vrednosti, ki poveže vrednost izdelka oz. njegovo ceno ter njegovo kakovost z ekonomskega vidika. Povezava obeh pojmov je nedvomna. Cilj pristopa je ugotoviti, ali je povezava sorazmerna ali obratno sorazmerna oz. ali je vložek v dvig kakovosti koristen. Pristop je subjektiven in težko merljiv. Največjo korist ima odjemalec, ki dobi kakovosten izdelek za razumno ceno.
- Pristop z vidika uporabnika, ki je najbolj tipičen pristop in kakovost smatra kot zmožnost, da izdelek zadovolji ali preseže pričakovanja kupca. Pristop je težaven, ker je subjektivno zaznavo kupca težko prevesti v merljive lastnosti izdelka. Končni cilj pristopa je ovrednotiti razkorak med pričakovanji ter zadovoljstvom kupca oz. uporabnika.

3.2 Planiranje kakovosti

Planiranje kakovosti je proces, ki vključuje identifikacijo za projekt ustreznih standardov kakovosti, izdelavo načrta kakovosti ter stalno skrb, da se planirana kakovost oz. standardi vgradijo v produkt. Planiranje kakovosti izvajamo vzporedno z drugimi aktivnostmi planiranja, saj ima tudi planiranje obsega, časa in stroškov komponento kakovosti (Fuller et al., 2008, str. 292).

3.2.1 Vhodi v proces planiranja kakovosti

Vhodi v proces planiranja kakovosti so povezani s trenutnim in preteklimi projekti ter v povezavi z organizacijskimi lastnostmi podjetja. Organizacijske lastnosti podjetja vključujejo faktorje, kot so politika glede kakovosti in pristop h kakovosti, ki je podjetju najbližji in najpomembnejši. Tudi procedure in uveljavljeni procesi, ki so v povezavi s

kakovostjo ter izkušnje in dognanja, ki sledijo iz izvajanja preteklih projektov. S kakovostjo povezani vhodi so tudi projektni dokumenti, ki opredeljujejo obseg in druge cilje projekta.

3.2.2 Tehnike planiranja kakovosti

Med planiranjem kakovosti uporabljane tehnike so analiza stroškov in koristi, primerjalno preverjanje (angl. *benchmarking*), uporaba eksperimentiranja (angl *Design of Experiments*), uporaba diagramov poteka, statistična obdelava podatkov itd.

Analiza stroškov in koristi vključuje analizo razmerja med stroški, ki nastanejo zaradi implementacije višje kakovosti v produkt, ter pozitivnimi učinki, ki jih bo višja kakovost produkta povzročila. V realnosti se lahko zgodi, da višja kakovost nima pomembnejše poslovne vrednosti. Zato je potrebno izvesti analizo, kdaj ima višja kakovost pozitivne posledice, npr. na prodajo produkta in posledično na povečane prihodke iz prodaje. Tehnika vključuje kalkulacije, kot je, npr. ROI (angl. *Return of Investement*), ki odločevalcem predstavi vrsto alternativ pri odločanju o kakovosti produkta. Hkrati je omejena, ker vse vidike kakovosti prehitro postavi na finančno osnovo (Fuller et al., 2008, str. 294). Analiza pomeni analizo stroškov, ki bi nastali, da bi bil končni rezultat projekta ustrezne kakovosti. Torej stroške, ki jih kakovost zahteva in stroške, ki bodo nastali, če bo kakovost neustrezna. Stroški nekakovosti so odpravljanje napak, modifikacije produkta in stroški, ki bi nastali zaradi sodnih obveznosti ali tožb (Fuller et al., 2008, str. 296).

Primerjalno preverjanje je tehnika, ki zajema analizo podobnih konkurenčnih izdelkov ali storitev, ki so na trgu in s katerimi izdelek oz. storitev tekmuje za tržni delež. Namen tehnike je dohiteti ali prehiteti konkurenco ter njene produkte z vidika kakovosti. Ker tudi nivo procesov, ki tečejo v podjetju vpliva na kakovost produktov, v namen primerjanja organiziranosti organizacije, uporabljamo tehniko za ocenjevanje zrelostnih stopenj podjetja angl. *Capability Maturity Model Integration* (v nadaljevanju CMMI), ki je uveljavljena predvsem na področju informacijske tehnologije.

Uporaba eksperimentiranja pomeni simuliranje sistemov z uporabo modela z vhodi, ki vplivajo na izhode sistema. Gre za zbiranje informacij z uporabo statističnih metod. Uporaba tehnike da odgovore na vprašanja, kot npr., kaj so ključni faktorji sistema, katere kombinacije faktorjev bodo dale optimalne rezultate, kakšne so glavne interakcije med faktorji sistema in katere kombinacije bodo pripomogle k manjši variaciji rezultatov. Tehnika predvideva identifikacijo ključnih faktorjev, iterativno eksperimentiranje z različnimi kombinacijami faktorjev ter model, ki upošteva vplive faktorjev na izhode iz sistema.

3.2.3 Izhodi iz procesa planiranja kakovosti

Izhodi iz procesa planiranja kakovosti vključujejo načrt managementa kakovosti, metrike kakovosti in seznam kakovostnih ciljev (angl. *quality checklists*), načrt izboljšav procesov,

izhodiščno kakovost (angl. *quality baseline*) ter v skladu s planiranimi cilji tudi načrt managementa projekta, kjer se bodo ti cilji odrazili (PMI, 2004, str. 200).

Načrt managementa kakovosti je podroben in obsežen dokument, ki določa, kako bodo ukrepi uvedeni v aktivnosti projekta. Dokument ne določa politike kakovosti, ampak se posveča konkretnim predvidenim situacijam. Načrt managementa kakovosti je na projektu uporabljen tudi kot vhod v druge procese planiranja (Fuller et al., 2008, str. 296).

Metrike kakovosti se uporabijo kot merilo kakovosti produkta ali procesov na operativnem nivoju. Podobno so seznami ciljev uporabljeni za preverjanje ali so bili izvedeni vsi predvideni ukrepi za dvig kakovosti ter izvedeni na pravi način.

Načrt izboljšave procesov je uporabljen za identifikacijo procesov, ki prispevajo h kakovosti in procesov, ki za kakovost nimajo pomena. Potreben je pristop, ki vključuje identifikacijo mej procesov, torej, kje se določen proces začne in kje se konča, kaj je njegov namen, kaj so vhodi in kaj izhodi ter kdo so nosilci procesa, analizo povezav med procesi, uporabo metrik za kontrolo procesov ter opredeljene cilje za izboljšanje le-teh.

Izhodiščna kakovost bo služila kot merilo kakovosti celotnega projekta. Izhodišče je lahko postavljeno glede na pretekle izkušnje s podobnih projektov ali pa je postavljeno s strani zunanjih ali notranjih strokovnjakov s področja. Tipični kakovostni cilji, ki jih opredeli izhodiščna kakovost, so število dovoljenih odpovedi produkta, uporabnost produkta, uporabniška izkušnja ipd.

3.3 Zagotavljanje kakovosti

Zagotavljanje kakovosti je sistematičen proces, ki vključuje definiranje, planiranje, izvajanje ter ocenjevanje managerskega procesa v samem podjetju. Cilj procesa je cilje kakovosti, ki so bili opredeljeni v procesu planiranja, uvesti v prakso. Vse aktivnosti so usmerjene k cilju, da se pridobi znanje ter zaupanje, da bo produkt ponovljivo ustrezno izdelan in izdelan v skladu z zahtevami. Proces zagotavljanja kakovosti bo izkoriščal v procesu kontrole kakovosti pridobljene informacije (PMI, 2004, str. 201).

Naslednji cilj zagotavljanja kakovosti je narediti produkt, ki je ustrezen za uporabo z vidika uporabnika ter ta cilj izpolniti v prvem poskusu. V praksi to pomeni izdelek z ustreznim obsegom funkcionalnosti in funkcionalnostmi brez napak, ki bi bistveno vplivale na delovanje produkta (Kelemen, 2003, str. 20).

Zagotavljanje kakovosti v praksi pomeni upravljanje z materiali in komponentami, sestavljanjem delov, upravljanje s storitvami povezanimi s produktom ter upravljanje s proizvodnimi ter razvojnimi procesi povezanimi s kakovostjo.

Podjetje, ki se zaveda pomena zagotavljanja kakovosti, ustanovi oddelek katerega funkcija je zagotavljanje kakovosti. Podjetje ima na voljo znanje in veščine ter standarde, za katere

mora oddelek poskrbeti, da so uvedeni v prakso. Posledično lahko podjetje pričakuje, da mu bo uspelo trgu ponuditi produkt, ki bo ustrezen ter hkrati imel konkurenčno ceno. Posebno pomembno vlogo pri zagotavljanju kakovosti nosi vrhnji management podjetja (Schwalbe, 2010, str. 298). Na vprašanje, kdaj podjetje doseže želeni cilj, torej ustrezno kakovost, pa odgovorijo odjemalci podjetja in ne podjetje samo.

3.3.1 Vhodi v proces zagotavljanja kakovosti

Vhodi v proces zagotavljanja kakovosti so načrt managementa kakovosti, metrike kakovosti, načrt za izboljšanje procesov, informacije o napredovanju projekta, zahteve za spremembe na projektu, poročila iz procesa kontrole kakovosti, ipd (Fuller et al, 2008, str. 299). Načrt managementa kakovosti je pripravljen že v fazi planiranja kakovosti in je uporabljen za implementacijo aktivnosti zagotavljanja kakovosti med izvajanjem vseh naslednjih faz projekta.

Informacije o napredovanju del na projektu ter poročila iz procesa kontrole kakovosti so, npr. poročila o rezultatih opravljenih testov in morajo biti v obliki, ki je primerna za nadaljnjo analizo, katere cilj je razumeti ali procesi zagotavljanja kakovosti učinkujejo (Fuller et al., 2008, str. 299).

3.3.2 Orodja in tehnike zagotavljanja kakovosti

Orodja namenjena procesu zagotavljanja kakovosti so identična tistim, uporabljenim v procesu planiranja kakovosti. V procesu zagotavljanja kakovosti pa dodatno prideta do izraza dve novi orodji, ki sta procesna analiza ter presoja kakovosti (angl. *quality audit*). Presoja kakovosti je aktivnost, ki jo izvaja notranji presojevalec ali zunanje podjetje ali organizacija in vključuje pregled s kakovostjo povezanih procesov podjetja in ocenjuje, v kolikšni meri se v procesih uporabljajo uveljavljene najboljše prakse ter znanja pridobljena na preteklih projektih.

3.3.3 Izhodi iz procesa zagotavljanja kakovosti

Izhodi iz procesa zagotavljanja kakovosti so ukrepi, ki bodo pripomogli k dvigu kakovosti produkta. Tipično gre za spremembe v poteku aktivnosti in poteku procesov, priporočene korektivne ukrepe na nivoju podjetja ipd. Vse spremembe se odrazijo tudi v spremembah projektne dokumentacije, ki je, spremenjena, eden izmed izhodov iz procesa (PMI, 2004, str. 202).

3.4 Kontrola kakovosti

Kontrola kakovosti je proces, katerega namen je, preko opazovanja in merjenja lastnosti produkta, ugotoviti, ali je ta dosegel predvideno stopnjo kakovosti oz. bil izdelan v skladu s standardi. Kadar temu ni tako, je cilj procesa najti načine za eliminacijo neskladnosti (PMI, 2004, str. 206).

Proces kontrole kakovosti se izvaja med samimi procesi ali na koncu procesa izdelave produkta. V preteklosti je bil večji poudarek na kontroli kakovosti ob koncu procesa izdelave. S testiranjem in inšpekcijami naj bi se preprečilo, da bi se produkt z napakami pojavil na trgu (Kelemen, 2003, str. 22). Sodobnejši pristop vodi prepričanje, da se kakovosti v produkt na tak način ne da vgraditi ob koncu proizvodnega procesa. Slabo zasnovan produkt ne moremo narediti kakovosten. Fokus se je zato premaknil proti fazi zasnove produkta in v kontrolo vmesnih stopenj proizvodnega procesa, ter na področje preprečevanja napak. Kontrolo kakovosti v podjetjih še vedno izvajajo specifični oddelki oz. skupine, ki imajo pooblastila ukrepati s ciljem odprave vzrokov za neskladnosti s kakovostnimi cilji.

Vhodi v proces kontrole kakovosti so (PMI, 2004, str. 206):

- plan managementa projekta,
- metrike kakovosti,
- seznam kakovostnih ciljev,
- poročila o opravljenem delu,
- organizacijske lastnosti podjetja,
- odobrene spremembe ter
- seznam predvidenih rezultatov projekta.

3.4.1 Orodja za kontrolo kakovosti

Orodja za izvajanje kontrole kakovosti so številna. Sedem temeljnih orodij za kontrolo kakovosti je diagram vzrokov in posledic (angl. *cause and effect diagram*), kontrolne karte (angl. *control chart*), histogram, Paretova karta, diagram poteka (angl. *run chart*), razsevni graf (angl. *scatter diagram*), inšpekcija ter testiranje. Izjemno pomembno orodje za kontrolo kakovosti na področju razvoja programske opreme je ravno testiranje (Schwalbe, 2010, str. 300), ki bo podrobneje opisano v posebnem poglavju.

3.4.2 Izhodi iz procesa kontrole kakovosti

Med ključne izhode kontrole kakovosti štejemo izmerjeno kakovost produkta. Drugi izhodi iz procesa kontrole kakovosti so še odločitve o sprejemljivosti, identifikacija potreb po spremembi procesov izdelave produkta ter odločitve o potrebi po ponovitvi določenih aktivnosti, zaradi neustreznih lastnosti produkta (Schwalbe, 2010, str. 299). Napake na produktu oz. neskladnost s specifikacijami običajno odkrijemo z uporabo orodij za kontrolo kakovosti, izmed katerih je največkrat v uporabi že omenjeni proces testiranja.

Odločitev o sprejemljivosti produkta je sprejeta po pregledu produkta s strani deležnikov projekta. V primeru, da je produkt sprejemljiv za vse deležnike projekta, obvelja, da gre za potrjen produkt. V nasprotnem primeru je potrebno ponovno delo na nekaterih aktivnostih (Schwalbe, 2010, str. 300). Zaradi ponovnega dela pride na projektih do prekoračitve časovnih ter stroškovnih okvirjev. Zato podjetja težijo k cilju, da bi bilo na projektih čim

manj ponovnega dela, kar zagotovijo z vpeljavo funkcije managementa kakovosti. Torej, uspešna podjetja kakovost planirajo, zagotavljajo ter kontrolirajo.

Tretji rezultat kontrole kakovosti je identifikacija potreb po spremembah in prilagoditvah procesa izdelave produkta. Težimo k temu, da bi v prihodnosti odpravili ali sploh preprečili, v času kontrole, zaznane napake v procesu, ki vplivajo na kakovost. Rezultat kontrole kakovosti so med drugim tudi prilagojeni projektni dokumenti in načrti izvajanja projektnih aktivnosti (PMI, 2004, str. 214).

4 KAKOVOST PROGRAMSKE OPREME

4.1 Definicija kakovosti programske opreme

Kakovost je v splošnem težko opredeliti, saj so za različne ocenjevalce pomembni različni vidiki kakovosti, ki so opisani v poglavju 3.1. Kakovost programske opreme, kot tehnična kategorija, pa je natančno opredeljena z modeli kakovosti, ki so sestavljeni iz skupin faktorjev, ki predstavljajo različne atribute kakovosti programske opreme. V uporabi je več modelov, ki so bili definirani s strani različnih organizacij ter strokovnjakov s področja razvoja programske opreme.

Znani model faktorjev kakovosti, imenovan McCallov model sestavlja 11 faktorjev, ki so razporejeni v 3 skupine, glede na sorodnost. Kasneje so bili predlagani tudi sodobnejši modeli, npr. Deutsch ter Willisov model, Evans ter Marchiniakov model, vendar se ti modeli ne razlikujejo bistveno od v nadaljevanju opisanega McCallovega modela (Galin, 2004, str. 37).

4.2 McCallov model kakovosti programske opreme

McCallov model sestavljajo 3 skupine faktorjev, ki zajemajo tako zunanje oz. uporabniške, kot tudi bolj tehnične, notranje faktorje kakovosti. Skupine opisane v nadaljevanju so (Galin, 2004, str. 37):

- skupina operativnosti programske opreme,
- skupina vzdrževalnosti programske opreme ter
- skupina prenosljivosti programske opreme.

4.2.1 Skupina operativnosti programske opreme

Skupino sestavlja 5 faktorjev, ki opredelijo uporabno vrednost programske rešitve oz. njeno vrednost za uporabnika:

- Ustreznost. Faktor ustreznosti govori o tem, v kakšni meri programska rešitev deluje skladno z definiranimi zahtevami, ki specificirajo delovanje programske rešitve oz. izhode iz rešitve glede na vhode.

- Zanesljivost. Faktor se nanaša na zmožnost programske rešitve, da zagotovi rezultate. Opredeli največje število napak na enoto delovanja programske opreme, gledano v celoti ali samo za posamezen modul. Visok faktor kakovosti pomeni njeno solidno ter odporno zgradbo. Zanesljivost se kaže v tveganju za odpoved in številu odpovedi programske opreme. Cilj spremljanja zanesljivosti je v zmanjšanju ter preprečevanju časa, ko programska oprema ni na voljo, deluje z napakami ali sploh ne deluje. Zanesljivost neposredno zadeva uporabnika ter njegovo poslovanje.
- Učinkovitost. Kakovost izvorne kode ter arhitektura programske rešitve sta elementa, ki zagotavljata zmogljivost programske opreme med delovanjem, torej njeno učinkovitost. Učinkovitost je posebej pomembna pri aplikacijah, ki zahtevajo visoko hitrost izvajanja, npr. algoritmično procesiranje ali procesiranje transakcij. Neučinkovita programska oprema je počasna, neodzivna in potencialno pomeni problem za uporabnika rešitve.
- Zaščita. Faktor zaščite programske opreme je stopnja verjetnosti, da pride do vdora v delovanje programske rešitve s strani nepooblaščenih subjektov. Vzroki za vdore so v slabem kodiranju ali slabi arhitekturi programske rešitve. Posledice se kažejo v oteženem ali prekinjenem delovanju programske opreme ter okrnjenem poslovanju uporabnika.
- Uporabnost. Faktor uporabnosti govori o številu osebja, ki ga naročnik programske rešitve potrebuje za začetek uporabe preko usposabljanja, ter je v povezavi s časom, ki je potreben, da se uporabnik nauči rešitev uporabljati.

4.2.2 Skupina vzdrževalnosti programske opreme

Skupina vzdrževalnosti ima 3 faktorje. Vzdrževanje je opredeljeno kot vzdrževanje, da bi se doseglo pravilno delovanje rešitve, vzdrževanje, ki zajema modifikacije za ustreznejše delovanje ter vzdrževanje z namenom, da se delovanje optimizira. Faktorji vzdrževalnosti so:

- Vzdrževalnost. Faktor opredeli potreben napor, da bi se obstoječa programska rešitev modificirala, ko je ugotovljeno neustrezno delovanje programske rešitve. Pri tem je v napor vključen tudi čas iskanja vzroka za napačno delovanje in tudi čas za verifikacijo ter validacijo ustreznosti rešitve.
- Prilagodljivost. Faktor prilagodljivosti pomeni zmožnost, da se programsko opremo prilagodi potrebam različnih strank s kar najmanjšo izrabo virov. Pomeni tudi zmožnost, da se programska oprema nadgradi, da bi zadostila novim in drugačnim poslovnim potrebam ter okolju.
- Testabilnost. Testabilnost oz. pripravljenost na testiranje, opredelimo kot faktor, ki pove v kakšnem obsegu obstoječa programska oprema omogoča testerju potrditi ustrezno delovanje preko vmesnih rezultatov operacij ali zapisovanja dogodkov v testne datoteke. Testabilnost je večja, če programska oprema vključuje samodejno diagnosticiranje napačnega delovanja, npr. pred samim zagonom. Samo-

diagnosticiranje ali drugo diagnosticiranje je v pomoč tudi vzdrževalcem programske opreme.

Vzdrževanje pomeni tudi zmožnost za prilagoditev in prenos vzdrževanja programske opreme med razvojnimi ekipami. Merjenje ter spremljanje faktorja vzdrževanja ima velik pomen pri sistemih, kjer je potrebno spremembe hitro ponuditi uporabnikom ter tam, kjer se poslovno okolje uporabnikov programske opreme spreminja dinamično. Tudi v tem primeru je odzivnost ponudnikov programske opreme pomembna. Pomembni pa so tudi stroški vzdrževanja, ki morajo biti čim nižji ter kontrolirani.

Na zmožnost vzdrževanja v veliki meri vpliva velikost programske opreme merjena kot število vrstic izvorne kode in ima neposreden vpliv na njeno vzdrževanje. Število vrstic izvorne kode je ponekod v uporabi kot merilo produktivnosti razvojnih ekip, kar ne prispeva k težnji pisati kodo modularno, s funkcijami, ki bi bile večkrat uporabljane v več programskih komponentah. Posledično bi bilo vzdrževanje učinkovitejše.

4.2.3 Skupina prenosljivosti programske opreme

Skupina definira zmožnost, da se programske module, ki sestavljajo programsko opremo, uporabi v drugih okoljih, na drugačnih strojnih platformah, operacijskih sistemih in zmožnost, da se jih uporabi v prihodnjih produktih. Skupino prenosljivosti sestavljajo:

- Prenosljivost. Faktor, ki opredeljuje zmožnost, da se programsko opremo prilagodi in nato enakovredno uporablja na drugih strojnih platformah, pod drugačnim operacijskim sistemom ipd.
- Ponovna uporabnost modulov. Faktor opredeljuje zmožnost, da se posamezne programske module nespremenjene uporablja kot del različnih in prihodnjih programskih rešitev. Predpogoj za visoko stopnjo ponovne uporabnosti je smiselno členjenje funkcij ter ustrezna arhitektura programske opreme. Ponovna uporabnost modulov bo v prihodnosti pomenila hitrejši razvoj nove programske opreme ter nižje stroške razvoja. Pričakujemo tudi višjo kakovost, ker so bile napake v programskih modulih zaznane ter odpravljene že med predhodnimi projekti.
- Zamenljivost. Faktor, ki opredeljuje zmožnost ali je moč programsko opremo brez prilagoditev enakovredno uporabljati na drugih strojnih platformah, pod drugačnimi operacijskimi sistemi ipd. V ospredju je tudi zmožnost, da se rezultate programske rešitve uporabi kot vhod v programsko opremo na drugem sistemu.

4.3 Standardi kakovosti s področja programske opreme

Standardi kakovosti so dokumenti, ki pokrivajo najrazličnejša področja delovanja podjetij in njihova naloga je podjetjem zagotoviti ekonomske, tehnološke ter družbene koristi. Tehnološke koristi se kažejo v premagovanju tehničnih ovir, ki pripomorejo h kakovostnejši ter učinkovitejši ponudbi izdelkov, iz katere sledi ekonomska vrednost.

Podjetja, ki so vpeljala standarde kakovosti, imajo učinkovitejše delovanje in manj stroškov, lažje povečajo produktivnost, lažje osvajajo nova tržišča, imajo zadovoljnejše stranke ter njihovo poslovanje manj negativno vpliva na okolje, v katerem se nahajajo. Podjetjem predstavljajo strateško orodje in priporočila, ki omogočajo konkurenčnost na modernem trgu.

Domena izdajanja standardov je v rokah več organizacij, med katerimi sta vodilni organizacija ISO (angl. *International Standard Organization*, v nadaljevanju ISO), ki s standardi sodeluje v širokem spektru področij in panog ter bolj tehnološko usmerjena organizacija IEEE (angl. *Institute of Electrical and Electronics Engineers*, v nadaljevanju IEEE).

ISO standardi s področja elektrotehnike nosijo oznako ISO/IEC (angl. *International Electrotechnical Commission*). Standardi organizacije so, skozi proces usklajevanja, razviti s strani uporabnikov samih. Pri tem sodelujejo strokovnjaki izbranega področja s celega sveta. Na ta način se skozi razvite standarde zrcalijo mednarodne izkušnje ter znanje. Razvoj ISO standardov gre v skladu s štirimi osnovnimi principi (ISO/IEC, 2013):

- ISO standardi so odgovor na potrebe trga. Organizacija ISO se odzove na potrebe industrije ali uporabnikov izdelkov in storitev.
- ISO standardi bazirajo na globalnem znanju ter izkušnjah. Strokovnjaki, ki razvijajo standarde, prihajajo s celotnega sveta in so formirani v oddelke t. i. strokovne komiteje. Strokovnjaki tehtajo in usklajujejo vse vidike standarda, obseg, definicijo ter vsebino.
- ISO standardi so razviti v skladu z vsemi deležniki, ki so, npr. strokovnjaki s področja, uporabniki, stranke in njihova podjetja, akademska sfera ter vladne organizacije.
- ISO standardi so razviti skozi proces usklajevanja ter zagotavljanja široke podpore. Vsa mnenja deležnikov so upoštevana.

Druga velika organizacija s področja razvoja standardov s področja tehnologije je IEEE. Organizacija ima svoje korenine v 19. stoletju in je sprva združevala strokovnjake elektrotehničnega področja. Danes ima organizacija IEEE prek 395.000 članov v 160 državah sveta in je globalna sila na področju tehnologije. Enako kot se je širilo članstvo, so se širila tudi področja, s katerimi se ukvarja organizacija. Od primarnega področja elektrotehnike so se področja širila na polje računalniške ter tudi drugih tehnologij, robotike, ultrazvoka ipd. Polno ime se iz tega razloga uporablja le za formalne namene, sicer pa je v uporabi kratica IEEE.

V nadaljevanju bodo predstavljena priporočila ter standardi, ki so povezani s kakovostjo ter razvojem programske opreme v različnih fazah življenjskega cikla projekta. Opisani standardi opredelijo faktorje kakovosti, definirajo razvojni proces in razvojne aktivnosti, usmerjajo pripravo projektnih dokumentov, testnih načrtov ter poročil.

4.3.1 ISO 9126

Sodelovanje strokovnjakov s področja programske opreme, pod pokroviteljstvom organizacije ISO, je vodilo v nastanek dokumenta oz. standarda z oznako ISO 9126 s polnim imenom angl. *Software Product Quality – Quality Model*, ki obravnava kakovost programske opreme. Dokument definira 6 področij oz. kategorij kakovosti programske opreme, ki so (Naik & Triphathy, 2008, str. 530):

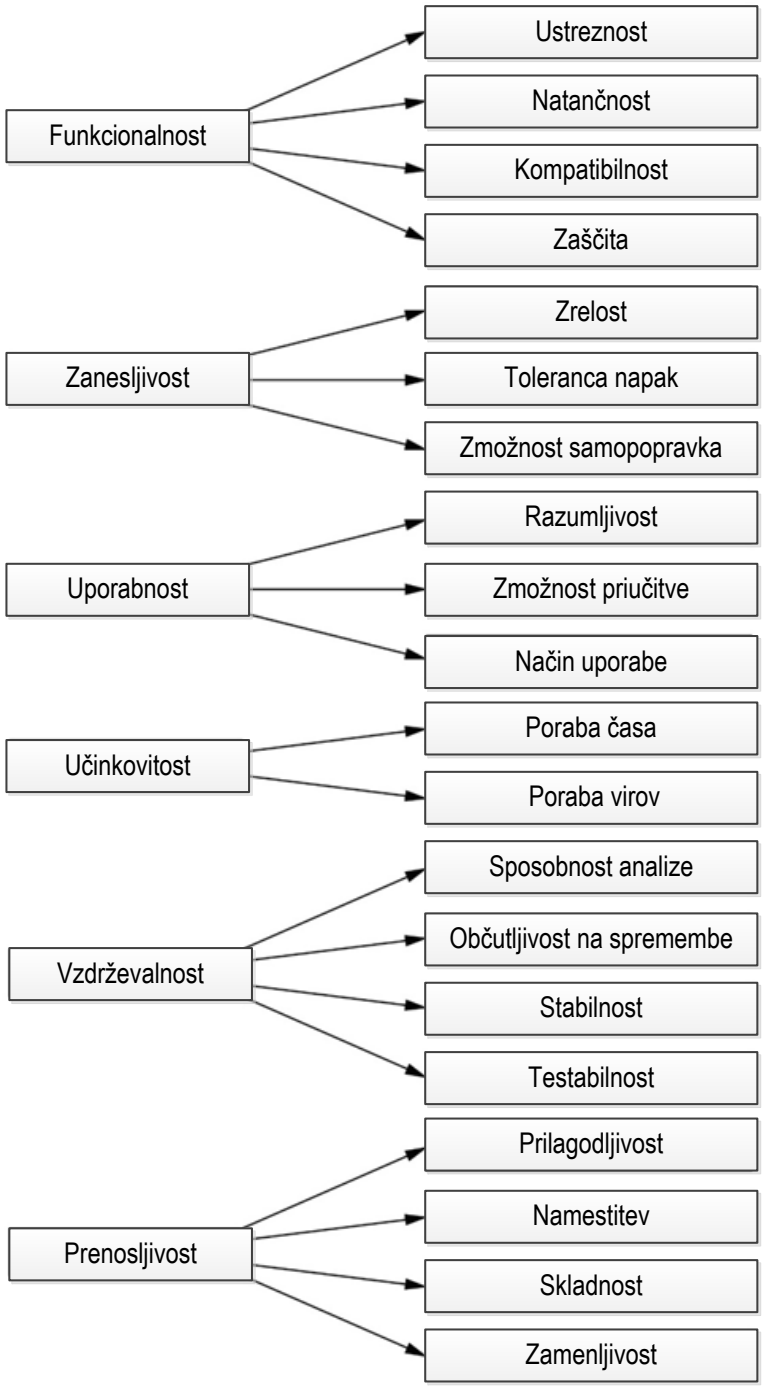
- funkcionalnost (angl. *Functionality*),
- zanesljivost (angl. *Reliability*),
- uporabnost (angl. Usability),
- učinkovitost (angl. Efficiency),
- vzdrževalnost (angl. Maintability),
- prenosljivost (angl. *Portability*).

Posamezne kategorije ter pod-kategorije imajo svoj par v McCallovem modelu kakovosti programske opreme. Pomeni se v obeh modelih praktično ne razlikujejo, obstaja le nekaj razlik, ki bodo opisana v nadaljevanju poglavja. Model kakovosti programske opreme po ISO 9126 razdeli posamezne kategorije v 20 pod-kategorij, ki jih prikazuje Slika 9:. Standard dopušča, da si dodatno razdelitev osnovnih kategorij, vsako podjetje prikroji po svoje, glede na razumevanje svojih potreb. Ko podjetje razume svoje potrebe in definira model, pa je pomembno, da teži k stalnim izboljšavam in doseganju višje stopnje posamezne pod-kategorije.

Bistvene razlike med McCallovim ter ISO 9126 modelom kakovosti programske opreme so (Naik & Tripathy, 2008, str. 534):

- ISO 9126 model se osredotoči na notranje in zunanje kategorije kakovosti ter kategorije, ki so vidne uporabniku (angl. *Quality in Use*), medtem ko se McCallov model posveča notranjim in bolj tehničnim vidikom kakovosti. Kakovost z uporabniškega vidika je posledica zunanje kakovosti, ki gradi na notranji kakovosti programske opreme.
- V McCallovem modelu je vpliv kakovostnega kriterija manj omejen in lahko vpliva na več faktorjev kakovosti, medtem ko v ISO 9126 pod-kategorija vpliva le na svojo izvorno kategorijo.
- Testabilnost, ki je v McCallovem modelu faktor višjega reda, je v ISO 9126 modelu pod-kategorija vzdrževalnosti.

Slika 9: Kategorije ter pod-kategorije kakovosti po standardu ISO 9126



Vir: S. Naik & P. Tripathy, Software Testing and Quality Assurance: Theory and Practice, 2008, str. 532.

4.3.2 ISO/IEC 12207

Prva izdaja standarda ISO 12207, s polnim imenom angl. *Systems and software engineering – Software life cycle processes*, sega v leto 1995 in standard je prvi, katerega namen je definirati faze, aktivnosti ter naloge s področja razvoja programske opreme, ki je bodisi del večjega sistema ali samostojen produkt. Standard je bil dopolnjen v letih 2002

ter 2004 in usklajen s standardoma ISO/IEC 15288 ter ISO/IEC 15504, obenem pa ohranja združljivost za nazaj. Zadnja verzija standarda nosi oznako ISO 12207:2008.

Glavni namen standarda je zagotoviti enoten pogled na naročilo, razvoj, uporabo in vzdrževanje programske opreme. Poenotenje standard doseže preko natančne definicije procesov v vseh življenjskih ciklih razvoja programske opreme ter uporabe enotnega izrazoslovja. Standard definira tudi vloge oz. odgovornosti za posamezne procese, aktivnosti in naloge.

Dodatno je standard opisan v nadaljevanju, saj identificira razvoju programske opreme specifične procese.

4.3.3 IEEE 829

IEEE 829 s polnim imenom angl. *IEEE Standard for Software and System Test Documentation* ima namen zagotoviti skupen okvir testnim procesom ter aktivnostim v vseh fazah življenjskega cikla procesa razvoja programske opreme, od faze zasnove do vzdrževanja programske opreme. Namen standarda je definirati testne aktivnosti ter vhode ter izhode iz aktivnosti ter določiti minimalen obseg testnih aktivnosti glede na stopnjo integritete (angl. *integrity level*) programske opreme.

Stopnja integritete je opredeljena kot resnost posledic, ki bi jih imela napaka v delovanju programske opreme na uporabnike. Standard predvideva 4 stopnje. Stopnja 4 pomeni napako, ki se lahko manifestira v izgubi življenja, posledicah za življenjsko okolje ipd. Medtem ko ima stopnja 1 za uporabnika zanemarljive posledice.

Glede na stopnjo integritete programske opreme standard opredeli tudi vrsto dokumentacije, ki je potrebna med procesom testiranja ter količino testnih aktivnosti. Opredeli tudi vsebino ter uporabo testne dokumentacije, tako krovne dokumentacije kot sta vrhnji testni načrt (angl. *Master Test Plan*) ter nivojski testni načrt (angl. *Level Test Plan*) ter opredeli vsebino in uporabo tudi povezanim dokumentom, kot so opis testnih primerov, testnih procedur, poročil ipd.

Glede na prejšnje izdaje je revizija iz leta 2008 bolj osredotočena na testiranje kot proces. Na ta način je standard usklajen s standardom ISO 12207, ki opredeljuje procese življenjskega cikla razvoja programske opreme.

4.3.4 IEEE 1016

IEEE 1016 s polnim imenom angl. *Systems Design – Software Design Description* je standard, ki opisuje pristop k opisovanju specifikacij programske opreme. Dokument predpisuje informacije, ki naj jih opis dizajna programske rešitve (angl. *Software Design Description,* v nadaljevanju SDD) vsebuje ter organizacijo dokumenta.

Standard je razdeljen v 5 poglavij, ki so splošen opis standarda ter kaj standard obsega, definicije uporabljenih izrazov, navodila za uporabo SDD dokumentov, opredelitev potrebne vsebine SDD dokumentov ter zadnji razdelek, ki opisuje priporočila za pripravo kakovostnih specifikacij. Dokument je usklajen s standardom ISO 12207. Ne predvideva izbranega modela razvoja programske opreme, temveč je s tega vidika univerzalen.

Najobsežnejši del IEEE 1016 standarda je opis, kaj je potrebno vzeti v obzir za izdelavo dobrih specifikacij programske opreme. Priporočilo predvideva, da je dokument sestavljen iz uvoda, splošnega funkcionalnega opisa programskega modula ter opisa posebnih zahtev za implementacijo. Standard priporoča za podroben opis delovanja programskega modula, uporabo modelirnega jezika UML (angl. *Unified Modeling Language*). Od osebe, ki sestavlja SDD dokument, zahteva fokus na izdelavi pravilnih, nedvoumnih, celovitih ter konsistentnih opisov oz. specifikacij, ki bodo osnova za kakovosten dizajn rešitve.

5 RAZVOJ PROGRAMSKE OPREME

Razvoj programske opreme je specifičen, iterativen proces. Vodimo ga projektno in projektni management je tisti, ki definira cilje projekta, obseg, čas, stroške ter kakovost. Prepoznamo faze projektnega managementa, fazo zasnove, planiranja, izvedbe, spremljanja in kontrole ter zaključevanja. Ločimo 3 osnovne modele življenjskega cikla razvoja (Peters, 2008, str. 105):

- Klasični fazni model. Življenjski cikel zasleduje tradicionalni pristop definicije zahtev, kreacije dizajna oz. rešitve, izvedbe dizajna, testiranja, izvedbe potrebnih izboljšav ter predaje končnega produkta. Slaba stran pristopa je, da niti stranka niti izvajalci ne vidijo rezultatov vse do zaključevanja projekta. To povečuje tveganje, saj se porabi znatna količina denarja ter opravi veliko delovnih ur do trenutka, ko je produkt predstavljen naročniku ter ga le-ta sprejme ali zavrne.
- Evolucijski model. Temelji na prepričanju, da se bodo med razvojem rešitve, zahteve spreminjale ter razvijale v naročniku vse bolj optimizirano obliko. Programsko opremo je namreč težko detajlno opredeliti že ob nastanku potrebe po njej. Med razvojem pride do kreacije novih idej, ki jih je v nekaterih primerih smiselno sprejeti in vgraditi v rešitev. Naročnik ima v tem modelu že kmalu na voljo prvi prototip, z omejeno funkcionalnostjo, ki mu omogoča zgoden odziv. Razvoj je hiter ter manj formalističen, brez jasno prepoznanih faz, kot so testiranje, dokumentiranje, presoja ujemanja specifikacij z zahtevami naročnika ipd. Projektni manager mora zato tesno sodelovati z razvojno ekipo ter tudi naročnikom, da obdrži kontrolo nad razvojnim procesom ter nastajajočim produktom. Pojavi se vprašanje, kdaj zaključiti tak projekt, saj se apetiti naročnika po funkcionalnostih ves čas večajo. Eden izmed pristopov je zaključiti projekt, ko so na projektu izčrpani viri glede odobrenih stroškov, razpoložljivosti človeških virov ter trajanja projekta.
- Amorfni model. Je najmanj urejen in formalno opredeljen model. Temelji na trudu posameznih razvojnih inženirjev bolj kot na procesu, ki bi ga le-ti izvajali. Model je

težko kontrolirati. Običajen pristop je definirati zahteve ter definirati datum, ko mora biti produkt končan, samo izbiro izvedbe ter pot, kako bodo razvojni inženirji do rešitve prišli, pa prepustiti njihovi prosti presoji.

Planiranje omejitev projekta, obsega, časa ter stroškov je osnova za uspeh projekta in hkrati zagotovilo za kakovost programske opreme. Potrebno je zasnovati produkt, definirati obseg, izbrati projektno ekipo, analizirati potrebe glede virov, narediti strukturo WBS, izgraditi časovnico ter opraviti druge povezane aktivnosti. Faza izvedbe da procesu razvoja programske opreme specifičnost. Njen potek je odvisen od izbranega modela razvoja in vpliva na management časa, na število iteracij, na procese kot je testiranje ipd.

Obseg magistrskega dela ne dovoljuje podrobno obravnavo vseh vidikov in specifik procesa razvoja programske opreme. Iz tega razloga so v nadaljevanju obravnavana področja, ki so v neposredni povezavi z managementom kakovosti ter časa in so potrebna za analizo delovanja izbranega podjetja.

5.1 Opredelitev programske opreme

Ko govorimo o programski opremi, si predstavljamo kopico vrstic programske kode, katere predstavljajo program, ki opravlja za uporabnika koristno opravilo. Pa vendar, na vprašanje ali je dovolj, da se pri zagotavljanju kakovostne programske opreme osredotočimo le na kodo, odgovorimo z ne, ker je programska oprema več kot samo koda (Galin, 2004, str. 15). IEEE definicija programske opreme je:

Računalniški programi, procedure in pripadajoča dokumentacija ter podatki, ki se nanašajo na delovanje računalniškega sistema.

Elementi programske opreme so:

- računalniški program oz. koda,
- procedure, ki predpisujejo pravilno uporabo sistema,
- dokumentacija in
- podatki, potrebni za delovanje programa.

Za kakovosten produkt ter njegovo učinkovito vzdrževanje so torej potrebni vsi štirje elementi programske opreme. Program oz. koda je potrebna iz očitnega razloga, ker opravlja za uporabnika koristno opravilo. Procedure in procesi definirajo način uporabe programa ter določajo potrebna znanja oseb, katerim je programska oprema namenjena.

Dokumentacija je potrebna razvijalcem, uporabnikom ter vzdrževalcem. Razvojna dokumentacija obsega specifikacije, dizajn dokumente, testna poročila ter opise programov, ki omogočajo kakovostno komunikacijo med razvojnimi ekipami ter učinkovito delo. Uporabniška dokumentacija obsega opise pravilnih postopkov uporabe. Dokumentacija namenjena vzdrževanju programske opreme pa opisuje delovanje

posameznih programskih modulov. Njen namen je, da ekipi, ki je zadolžena za vzdrževanje, omogoči učinkovito odkrivanje napak oz. vzrokov za nepravilno delovanje ter možnost hitrega prilagajanja ter nadgradnje rešitev.

Za delovanje programa so potrebni podatki, ki nastopajo v vlogi vhodnih parametrov v program. Posebna skupina podatkov so testni podatki, ki omogočajo testiranje pravilnosti delovanja programa.

5.2 Procesi specifični razvoju programske opreme

Na področju razvoja programske opreme identificiramo procese, ki so področju specifični. Standard ISO 12207 identificira sistemske procese (angl. *System Life Cycle Processes*), ki so identični procesom v standardu ISO/IEC 15288, vendar prirejeni razvoju programske opreme. Med sistemske procese standard uvršča:

- Procese, ki formalno urejajo odnos med naročnikom ter ponudnikom programske opreme (angl. *Agreement Processes*).
- Procese iz organizacijskega vidika podjetja, ki omogočijo podjetju oz. ponudniku programske opreme zagon projekta preko opredelitve managementa življenjskega cikla projekta, managementa potrebnih virov, managementa kakovosti itd.
- Procese, ki so splošni projektnemu managementu, med katere spadajo proces planiranja, kontrole, proces odločanja, management tveganj, proces merjenja ter proces managementa informacij.
- Tehnične procese, katerih namen je preoblikovati zahteve naročnika v produkt, zagotoviti konsistentno reprodukcijo produkta, omogočiti njegovo uporabo ter umik produkta iz uporabe.

Skupine procesov, ki so specifične razvoju programske opreme (angl. *Software Specific Processes*), so (ISO, 2008, str. 14):

- procesi implementacije programske opreme (angl. *Software Implementation Processes*),
- implementaciji pomožni procesi (angl. *Software Support Processes*) ter
- procesi za omogočanje večkratne uporabe posameznih modulov programske opreme (angl. *Software Reuse Processes*).

Procesom v skupini implementacije programske opreme je cilj izdelava produkta, ki je opredeljen kot programska oprema. Produkt mora zadostiti zahtevam deležnikov projekta ter specifikacijam. Procesi nižjega nivoja v skupini implementacije so, poleg implementacije, še analiza zahtev, dizajn arhitekture programske opreme, detajlen dizajn programske opreme, izgradnja programske opreme, integracija ter kvalitativno testiranje.

Strategija implementacije predvideva prosto izbiro modela razvoja programske opreme, razen v primerih, ko je izbor formalno določen, npr. s pogodbo z naročnikom.

Proces kvalitativnega testiranja predvideva verifikacijo, da je programska oprema v skladu s specifikacijami, preko izhodov iz procesa, ki so:

- razviti kriteriji za verifikacijo, ki bodo nedvomno potrdili ustreznost,
- opravljena verifikacija v skladu z razvitimi kriteriji,
- zabeleženi testni rezultati ter
- razvita strategija regresijskega testiranja, ko so na produktu opravljene modifikacije.

Implementaciji pomožni procesi (angl. *Software Support Processes*) so procesi, ki so v povezavi z vzdrževanem dokumentacije, vzdrževanjem konfiguracij, zagotavljanjem kakovosti programske opreme, verifikacije in validacije programskih modulov ter pregledov napredka in stanja razvijajoče programske opreme.

Proces zagotavljanja kakovosti je opredeljen kot proces, katerega naloga je zagotoviti produkt in proces njegove izdelave v skladu z načrtom kakovosti (ISO, 2008, str. 69). Izhodi iz procesa so strategija zagotavljanja kakovosti, formalni dokazi, da se proces izvaja ter vzdržuje, zabeležene neskladnosti z zahtevami ter verifikacija, da so procesi ter produkti skladni z uveljavljenimi standardi.

5.3 Modeli razvoja programske opreme

Modeli razvoja programske opreme so številni in njihova izbira ima neposreden vpliv na rabo časa že med fazo planiranja projekta, predvsem pa med fazo izvedbe. Izdelava programske opreme zahteva proces, ki se začne z identifikacijo problema, izdelavo kode oz. programske rešitve in predajo produkta v uporabo stranki. Različna podjetja in razvojne ekipe izberejo drugačen model razvoja programske opreme, glede na velikost, kulturo, zrelost in organizacijo podjetja, lastnosti projekta ter druge okoliščine. Vsak model ima prednosti ter slabosti in model, ki najbolje deluje, npr. na manjših in krajših projektih, ne bo enako dobro deloval na daljših projektih ali projektih z visoko stopnjo tveganja, npr. ko je predviden produkt projekta varnostno kritična programska rešitev.

Modeli, ki so v literaturi najpogosteje opisani so model velikega poka (angl. *Big Bang Model*), slapovni model (angl. *Waterfall Model*), spiralni model, evolucijski model ter vse popularnejši modeli na osnovi agilnih metod. V namen kasnejše analize izbranega podjetja bodo opisani nekateri modeli razvoja, ki so si po specifikah precej različni, vendar učinkovito ilustrirajo procese razvoja programske opreme.

Model velikega poka

Patton (2006, str. 31) opiše model kot model, kjer kritično število razvojnikov, dovolj velik proračun projekta in dovolj časa, ustvarita produkt, ali pa ga tudi ne. Model je skrajno tvegan, je neformalen, praktično brez planiranja časa in planiranega testiranja. Ves trud je posvečen izdelavi kode. Končni rok končanja projekta je fleksibilen, vendar se fleksibilnost pričakuje tudi od strank projekta. Prednost modela je njegova enostavnost.

Model Kodiraj in popravljaj

Model kodiraj in popravljaj (angl. *Code and Fix*) je nizko na evolucijski lestvici modelov in je običajna intuitivna izbira razvojnih ekip, če le-te zavestno ne izberejo drugačnega pristopa. Delovanje po tem modelu zahteva vsaj grobe specifikacije in predstavo, kaj naj ustvarjen produkt bo. Nadaljuje se z iteracijami kodiranja, testiranja ter odpravljanja hroščev. V nekem trenutku projektni manager ali odgovorna oseba odloči, da je bilo iteracij dovolj in izda produkt stranki oz. ponudi trgu.

Model deluje dobro za manjše projekte, saj je zaradi razmeroma malo formalnega dela na dokumentiranju ter dizajnu, možno relativno hitro pokazati prve rezultate projekta. Slaba lastnost pa je veliko ponovljenega dela ter dejstvo, da produkt praktično nikoli ni končan in je verjetnost, da je brez hroščev majhna, saj je bila zadnja razvojna iteracija prekinjena zaradi različnih vzrokov, npr. izteka časa. Kljub tem lastnostim, je bil model uporabljen v več večjih in znanih projektih razvoja programske opreme (Patton, 2006, str. 32).

Slapovni model

Model je urejen in sistematičen. Njegove faze so razdeljene na ločene aktivnosti, običajno na izdelavo zahtev, analizo, dizajn, razvoj oz. izvedbo, testiranje ter zaključek. Preden se ekipa loti nove faze se sestane ter evalvira, če je proces ustrezno nadaljevati ali pa je potrebno opraviti manjkajoče ali pomanjkljivo opravljene aktivnosti v trenutni fazi.

Slapovni model ima 3 pomembne lastnosti oz. omejitve (Patton, 2006, str. 33):

- Poudarek je na izdelavi dokumentacije ter dizajnu dokumentov, ki opredeljujejo, kaj naj produkt dela in na kakšne načine bodo izvedene morebitne modifikacije. Samo kodiranje predstavlja le eno aktivnost v diagramu slapovnega modela.
- Faze se med seboj praviloma ne prekrivajo in se izvajajo v zaporedju.
- Prehodi med fazami so praviloma možni le v smeri naprej.

Lastnosti modela so omejujoče, vendar delujejo ustrezno, ko gre za dobro premišljene specifikacije in je razvojna ekipa dovolj zrela in disciplinirana. Praviloma je potrebno definirati praktično vse lastnosti projekta, še preden je napisana prva vrstica kode. Ravno to je glavna pomanjkljivost modela, saj preteče relativno veliko časa, preden je moč predstaviti kakršenkoli prototip produkta. Model je tudi manj primeren za iterativen razvoj in iterativne miselne vzorce, s katerimi ljudje rešujemo kompleksne probleme (Peters, 2008, str. 109).

Model je pomemben, ker opredeljuje testiranje kot samostojno enoto. Torej napeljuje k uporabi orodij za zagotavljanje in kontrolo kakovosti. Prav tako eksplicitno zahteva, da je v fazi planiranja časa, testnim aktivnostim dodeljen čas, vendar ga model dodeljuje na koncu, pred izdajo produkta.

Agilne metode

Agilne metode povzemajo najboljše prakse iz več modelov ter najboljše programerske prakse. Razvoj v skladu z agilnimi metodami vključuje stalno interakcijo s stranko, v nekaterih primerih z vključitvijo njenega predstavnika v razvojno ekipo. V primerjavi z drugimi modeli se zdi metoda nova in radikalna, vendar je v uporabi že zadnjih 30 let (Peters, 2008, str. 116). V zadnjem času pridobiva na popularnosti na področju manjših ter srednje velikih projektov. Model omogoča hitro pridobivanje povratnih informacij ter omogoča iterativno reševanje problemov.

Celotno delo na projektu je razdeljeno na aktivnosti oz. t. i. zgodbe (angl. *stories*), ki so ocenjene glede na kompleksnost in trajanje s točkami (angl. *story points*). Čas je razdeljen v enako dolge časovne intervale (angl. *sprint*), v katere se vstavi eno ali več aktivnosti s predvideno vsoto točk. Posamezne zgodbe niso daljše, kot je trajanje enega časovnega intervala.

V vsakem intervalu je želja končati vse predvidene zgodbe in imeti ob koncu vsakega intervala stabilen, delujoč produkt. V primeru, da posamezne zgodbe niso zaključene, se jih konča v naslednjem intervalu. Ekipa planira in analizira svoje delo ob vsakem začetku oz. koncu intervala ter uporablja razmerje med končanimi in planirani točkami kot merilo uspešnosti. Ugotovitve lahko prenese v ustrezne ukrepe že v naslednjem intervalu.

Med razvojem po agilnih metodah je prisotno stalno testiranje kode, ki se vrši v vsakem intervalu posebej na novo zgrajenih, modificiranih in odvisnih modulih.

Agilne metode imajo specifično lastnost, da med razvojem nimamo t. i. obdobja stabilnega produkta, ko se koda, ki se testira, ne bi spreminjala. Imamo torej stalno testiranje ter v istem času stalno spreminjanje kode, kar ni v skladu s pristopom, ki ga kot pogoj za uspešnost projektov predlaga Brooks (v Peters, 2008, str. 117). Pristop predvideva, da se spremembe vršijo le ob predvidenem času ter takrat več naenkrat in se ne dogajajo v fazi testiranja. Problem stabilnosti se rešuje z uporabo avtomatske izgradnje kode (angl. *daily* ali *nightly build*), stalnega testiranja in implementacije kodnih standardov.

Z vidika kakovosti so z uporabo agilnih metod zlahka vpelje prakso prenavljanja kode (angl. *refactoring)*, ki pomeni stalno izboljševanje posameznih programskih modulov ter posledično izboljšano kakovost kode. Dizajn posameznih modulov ostaja enostaven in modularen. V uporabi je tudi programiranje v paru (angl. *pair programming*), ko skupaj delujeta izkušenejši in manj izkušen razvojnik.

5.4 Zagotavljanje in kontrola kakovosti programske opreme

Zagotavljanje kakovosti programske opreme (angl. *Software Quality Assurance*) pomeni nadzorovati inženirske procese in načine, kako se programska koda razvija. Potreben je tako nadzor tehničnih vidikov razvoja, pretežno programiranja, kot tudi nadzor oz.

zagotavljanje ustrezne organiziranosti razvojne ekipe. Metode, s katerimi nadzorujemo razvoj, so različne, večkrat pa jih obravnavamo ter ocenjujemo z uporabo standardov ter modelov, kot npr. z družino ISO standardov ali z uporabo zrelostnih modelov, kot je CMMI.

Inženirske procese nadzoruje ekipa, ki je zadolžena za proces zagotavljanja kakovosti programske opreme. Za razliko od testne ekipe, ki testira programske rešitve in poroča o odkritih napakah oz. hroščih in na tak način pripomore h kakovosti programske opreme, je naloga ekipe za zagotavljanje kakovosti stalno izboljševanje procesa razvoja programske opreme, iskanje primernih orodij in metod in jih vpeljevati v razvojne aktivnosti (Patton, 2006, str. 333). Vpeljava orodij in metod mora povzročiti, da se hrošči v programski kodi sploh ne pojavijo oz. se v bistveno manjšem številu.

Podjetje, ki čuti potrebo po ekipi za zagotavljanje kakovosti, mora do nje priti postopno. Običajno ima podjetje sprva testno ekipo, ki preko testiranja kontrolira kakovost. Ko podjetje dozori, spozna, da se kakovosti ne da vsiliti v produkt s kontrolo, ampak je kakovost potrebno vgraditi s sistematičnim pristopom skozi vse življenjske faze projekta. Iz dela testne ekipe ustanovi ekipo za zagotavljanje kakovosti. V praksi to pomeni kakovostnejše specifikacije, planiranje časa, skrb za dokumentacijo in nenazadnje tudi testiranje že v fazi izvedbe. Vsi ti in tudi drugi procesi bodo prispevali k temu, da se hrošči v kodi ne bodo pojavili (Patton, 2006, str. 334).

Zagotavljanje kakovosti je načrtovan in sistematičen proces, s katerim spremljamo in skrbimo za kakovost v vseh fazah razvoja programske opreme. Posebnost managementa razvoja programske opreme je, da je management kakovosti praktično mogoč samo med zasnovo in izvedbo. Ko je na voljo prvi prototip, je le-ta lahko nespremenjen tudi za produkcijo. Programska oprema ni fizičen produkt. Pri fizičnih produktih, se korekcije izvaja tudi v fazi produkcije (Kelemen, 2003, str. 59).

V povezavi z zagotavljanjem kakovosti Galin (2004, str. 96) predvideva izdelavo dveh temeljnih dokumentov za zagotavljanje kakovosti programske opreme. To sta projektni razvojni načrt (angl. *Project Development Plan*) ter, s kakovostjo tesneje povezan, načrt kakovosti (angl. *Quality Plan*).

Projektni razvojni načrt

Projektni razvojni načrt je dokument, ki izhaja iz specifikacij projekta, začrta pot izvedbi projekta in upošteva vse dogovorjene omejitve projekta. Razvojni načrt je potreben zaradi skladnosti s standardi razvoja programske opreme ISO ter IEEE, prav tako pa je potreben po modelu CMMI za pridobitev zrelostne stopnje razvoja programske opreme (Galin, 2004, str. 96). Načrt navede predvidene rezultate projekta, ki so tipično:

- dokumenti, ki opisujejo končan produkt in uporabljene tehnološke rešitve, npr. dizajn dokumenti celotnega in posameznih modulov produkta,

- razvit produkt oz. programska rešitev ter
- usposabljanje uporabnikov programske rešitve.

Vsi rezultati projekta imajo poleg ostalih parametrov predviden tudi rok končanja. Načrt vsebuje izbrani pristop k managementu vsake posamezne faze ter določi predvidene standarde, metode in procedure. V načrtu je vsaki predvideni aktivnosti natančno določeno njeno trajanje, povezave do drugih aktivnosti ter potrebni izvajalci.

Rezultat planiranja je grafičen prikaz napredovanja projekta v obliki Gantt diagrama ali mrežnega diagrama, ki omogoča izdelavo časovnih analiz izvedbe projekta. Poleg omenjene vsebine projektni razvojni načrt vsebuje še pomembne mejnike projekta, seznam potrebne opreme, navede možna tveganja, kontrolne metode, in predvidene stroške (Galin, 2004, str. 100).

Načrt kakovosti

Načrt kakovosti je podroben dokument, katerega namen je zagotoviti, da se veščine in rezultati aktivnosti v predvidenem časovnem in stroškovnem okvirju prenesejo v kakovosten rezultat projekta, ki je skladen z zahtevami. Načrt kakovosti je izdelan med planiranjem in je lahko del projektnega razvojnega načrta ali pa je pripravljen kot samostojen dokument. Za obsežne projekte je načrt kakovosti razdeljen na v nadaljevanju navedene elemente. Načrt naj bo pregledan in odobren s strani deležnikov projekta in v skladu s smernicami kakovosti (Galin, 2004, str. 103). Elementi načrta kakovosti so:

- Kakovostni cilji, ki odražajo sprejemljive kakovostne kriterije produkta. Kakovostni cilji služijo kot referenca za oceno uspeha projekta z vidika kakovosti.
- Planirane aktivnosti ocenjevanja kakovosti. Tipično gre za pregled dizajn dokumentov, pregled kode, ipd. Za vsako aktivnost je definiran obseg pregleda, tip pregleda, kdaj med projektom se izvede ter odgovornosti.
- Pregled predvidenih testov. Definirano je kateri testi enot, integracijski ter sistemski testi, se bodo med potekom projekta izvajali.
- Predvidena časovnica testiranja, ki je skladna s splošno časovnico projekta.
- Odgovornosti in posebnosti testiranja.
- Predvideni prevzemni testi za programsko opremo, ki je razvita s strani zunanjih izvajalcev (angl. *outsourcing*).

Fazi planiranja sledi faza izvedbe, ki vključuje tehnične aktivnosti podrobnega dizajna, dokumentiranja, kodiranja in testiranja nastajajoče programske rešitve. Naloga projektnega managerja je koordiniranje razvojnih ekip, management stroškov in obsega projekta ter kontrola časovnice. Posredno je z vidika kakovosti pomembna kontrola časovnice oz. tem učinkovitejša raba časa, da bi imele razvojne ekipe na voljo dovolj časa za aktivnosti, ki so neposredno povezane s kakovostjo. Poleg strogo tehničnih disciplin je v ospredju testiranje, za katerega je predviden čas v vseh fazah razvoja, čeprav se intenziteta testnih

aktivnosti praviloma proti zaključku projekta povečuje. Testiranje kot pomemben del razvoja programske opreme je opisano v posebnem poglavju.

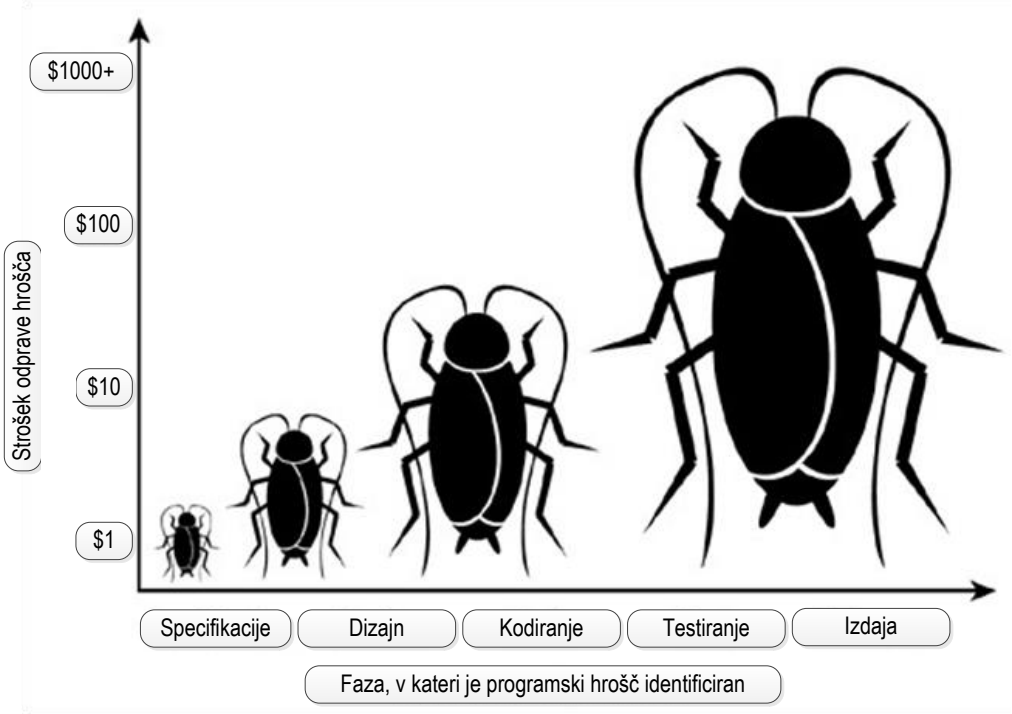
6 TESTIRANJE PROGRAMSKE OPREME

6.1 Testiranje kot orodje zagotavljanja in kontrole kakovosti

Testiranje programske opreme je pomemben del zagotavljanja in predvsem kontrole njene kakovosti. S tehničnega in funkcionalnega vidika je testiranje pomembno kot orodje za potrditev ustreznega delovanja programske opreme oz. za potrditev skladnosti z zahtevami in specifikacijami. S poslovnega vidika preprečuje negativne finančne posledice, saj si tako ponudnik programske opreme, kot tudi uporabnik ne moreta privoščiti uporabo pomanjkljive programske opreme. V takem primeru prihaja do nepredvidenih in nepotrebnih stroškov, ki so posledica odpovedi programske opreme med delovanjem, ter povezani s kasnejšim vzdrževanjem. Nekatere negativne posledice pa je težje ovrednotiti, npr. okrnjen ugled podjetja in izgubo potencialnih strank. V najslabšem primeru podjetje doleti celo tožba s strani uporabnikov programske rešitve in neposredne finančne posledice.

Ustrezno oz. celovito testiranje je zagotovilo, da bo programska oprema kakovostnejša in pripravljena za uporabo na trgu in bodo napake odkrite še v fazi razvoja, med testiranjem in ne kasneje, med uporabo. Predvsem odpravljanje hroščev je v slednji situaciji izjemno drag proces.

Slika 10: Naraščajoči stroški hrošča skozi faze razvoja programske opreme



Vir: R. Patton, Software Testing, 2006, str. 18.

Proces zagotavljanja in kontrole kakovosti je torej tesno povezan z različnimi vrstami testiranja. Kolikšen je obseg testiranja je odvisno od kompleksnosti razvijajoče programske opreme. Vendar praviloma časovno in glede človeških virov, zavzema kar polovico vseh razvojnih aktivnosti in tudi več, za varnostno kritične sisteme (Ammann & Offutt, 2008, str. 10) oz. vsaj 25 odstotkov (Živković, 2011, str. 30).

V kakšnem obsegu je programska oprema testirana je odvisno od velikosti testne ekipe, od izbire testov ter morebitne avtomatizacije testiranja. Kljub uporabi avtomatizacije in skrbni izbiri testov ter veliki testni ekipi, se moramo v praksi zadovoljiti z nepopolnim testiranjem. Zaradi kompleksnosti programskih rešitev in praktično neskončnega števila testnih primerov, je nemogoče testirati vse možne načine delovanja rešitve. Pomembna omejitev je čas, ki je vedno omejen.

6.2 Proces testiranja

Procesu testiranja identificiramo podobne faze kot projektu in ga je možno voditi kot pod-projekt znotraj projekta razvoja programske opreme. Testiranje v bolj grobi delitvi bi imelo 3 faze (Ammann & Offutt, 2008, str. 5):

- Pripravo oz. planiranje testiranja, ki obsega pripravo testnega načrta, dizajn testnih primerov ter pripravo opisov testnih procedur. Vključuje tudi razdeljevanje vlog in odgovornosti za posamezne testne nivoje in za celotno testiranje, planiranje in ocenjevanje časa potrebnega za izvedbo testov, poročanje ter predvidevanje različnih ukrepov glede na rezultate in izsledke testiranja.
- Izvedbo testiranja, ki poteka skladno s testnim načrtom. Poleg izvajanja testiranja in spremljanja poteka, je izjemno pomembna tudi sprotna administracija rezultatov testiranja, običajno v obliki poročil o identificiranih hroščih.
- Zaključevanje testiranja oz. predstavitev izsledkov ter končnih rezultatov testiranja. Faza je najzahtevnejša izmed vseh, ker je ključno, da zagotovimo koristnost rezultatov izvajanega testiranja, preko popravkov programskih modulov. Potrebno je zagotoviti, da se bodo v naslednji razvojni iteraciji hrošči odpravili in dvignilo nivo kakovosti programske opreme.

6.2.1 Priprava testiranja

Priprava testiranja, poleg drugih omenjenih aktivnosti, pomeni izdelavo testnega načrta, ki je testiranju bazični dokument, opisuje kaj in na kakšen način bo testirano ter omogoča kakovostno komunikacijo testerjev z razvojno ekipo. Z dokumentom testna ekipa sporoči razvojni ekipi svoje cilje in pričakovanja. Standard IEEE 829-1998 definira cilje testnega načrta (Patton, 2006, str. 264):

- predpisati obseg, pristop, vire in časovnico testnim aktivnostim,
- identificirati predmet testiranja, lastnosti, ki se testirajo ter
- opredeliti testne naloge, delegirati odgovornosti in identificirati tveganja.

Rezultat planiranja je torej testni načrt. Vendar je dokument predvsem stranski produkt miselnega procesa planiranja, ki je ključen za kasnejšo izvedbo testiranja in pomembnejše od samega dokumenta. Iz tega razloga je oblika in vsebina testnega načrta do neke mere drugačna za vsak posamezen projekt in podjetje (Patton, 2006, str. 264).

Planiranje testiranja programsko rešitev razdeli na specifične funkcionalnosti oz. module, ki jih posamezni člani ali več članov ekipe lahko testirajo ločeno. Planiranje v tej fazi še ne predpisuje pristopa k testiranju. Predpisuje oz. organizira le skupine testov glede na funkcionalnosti programske rešitve in jim dodeli identifikatorje, identificira posamezne funkcionalnosti ter pristop k testiranju ter strategijo. Planiranje testnih aktivnosti naslovi naslednje teme:

- Predpiše kriterije za primer neuspešnega rezultata testiranja in kriterije za primer uspešnega rezultata, kamor lahko štejemo tudi testiranje, ko je uspešnih predpisano število testnih primerov in ne prav vsi. Kriteriji so različni in odvisni od produkta, ki ga testiramo.
- Naslovi nekaj na videz samoumevnih tem, kot npr., kaj je predmet testiranja. Tako kot mora biti popolnoma jasno članom razvojne ekipe, kaj naj programska rešitev počne, mora biti enako poglobljeno tudi znanje testne ekipe. Opis produkta, ki je del testnega načrta izhaja iz dokumenta, ki opredeli delovanje naprave oz. iz specifikacij produkta.
- Opredeli kakovostne cilje produkta. Pri tem gre za cilje glede uporabnosti, števila funkcionalnosti, zanesljivosti produkta ter druge.
- Obsega razdelitev vlog in nalog ter odgovornosti članov testne ekipe.
- Opredeli dokumente, ki so rezultat testnih postopkov. Navede potrebno strojno in programsko opremo ter infrastrukturo za izvedbo testov.

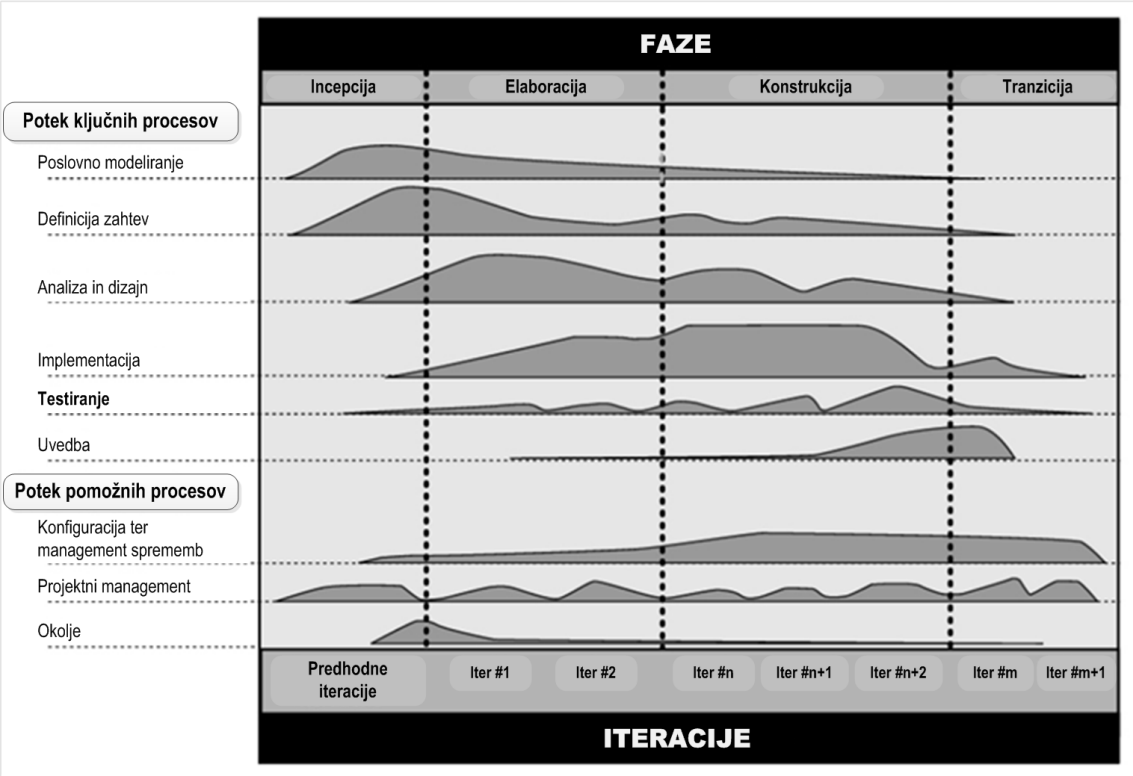
Naslednji cilj planiranja testiranja je opredelitev časovnih rokov za končanje posameznih faz testiranja. Pri tem je pomembna usklajenost celotne razvojne ekipe. V smislu učinkovite rabe časa je usklajenost ključna, če je izbrana strategija testiranja inkrementalna metoda testiranja. To pomeni, da se bodo testiranja izvajala z izgradnjo posameznih končanih modulov v končni produkt, v trenutku, ko bodo programski moduli na voljo.

V vsakem primeru velja, da testiranje ni enakomerno porazdeljeno skozi vse faze projekta. Intenziteta in število nalog naraščata proti koncu projekta in ključno je, da se na konec projekta razporedi zadostno število virov, da bi se časovni roki lahko izpolnili.

Slika 11 prikazuje intenziteto testiranja po IBM-ovi (angl. *International Business Machines*) metodi RUP (angl. *Rational Unified Process*), ki valovito narašča ob zaključevanju faze testiranja. Slika namiguje na iterativen razvoj programske opreme.

Dizajn in dokumentiranje testnih primerov je naslednji obsežen cilj planiranja testiranja. Testni primeri naj bodo podprti s podrobno definiranimi testnimi postopki, ki bodo v pomoč pri izvajanju ter ponovitvah testov.

Slika 11: Krivulja intenzivnosti testiranja v posameznih razvojnih fazah



Vir: Literate programming and the rational unified process, 2013.

6.2.2 Izvedba testiranja

Izvajanje testiranja v ožjem pomenu pomeni izvajanje testnih primerov, predvidenih v testnem načrtu, po opisanih testnih postopkih. Najpomembnejši rezultat izvajanja testnih postopkov bodo odkriti hrošči.

Nepravilno delovanje programske opreme, ki je posledica programskega hrošča, moramo pravilno prepoznati. Definicija programskega hrošča je (Patton, 2006, str. 267):

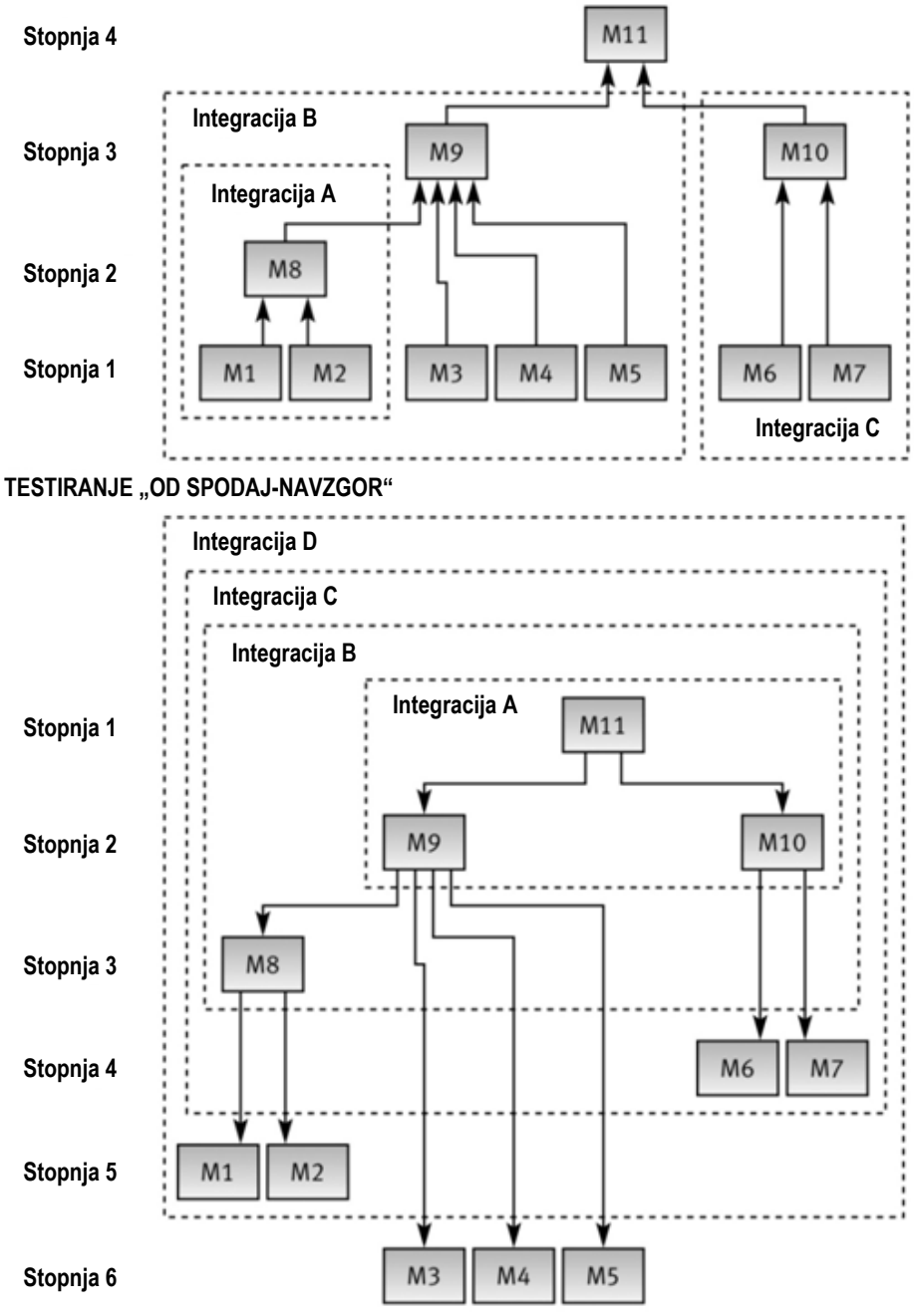
- Programska oprema ne naredi tistega, kar je v specifikacijah navedeno, da dela.
- Programska oprema naredi tisto, kar je navedeno v specifikacijah, da naj ne bi delala.
- Programska oprema naredi nekaj, kar v specifikacijah ni navedeno.
- Programska oprema ne naredi tistega, kar ni navedeno v specifikacijah, vendar bi moralo biti.

Z izvedbenega vidika lahko testiranje izvajamo na več načinov. Testiranje se lahko izvaja inkrementalno ali po t.i. metodi velikega poka ali v kombinaciji obeh pristopov (Galin, 2004, str. 180).

Metoda velikega poka pomeni, da so pred pričetkom testiranja vsi programski moduli končani in je potrebno potrditi njihovo delovanje kot celoto oz. potrditi delovanje končanega produkta. Osnova za testiranje so funkcionalne zahteve ter druge specifikacije

produkta. Testna ekipa iz funkcionalnih specifikacij razvije testni načrt, ki se izvaja na potencialno končanem produktu.

Slika 12: Prikaz strategij testiranja »od spodaj-navzgor« ter »od zgoraj-navzdol«



Vir: D. Galin, Software Quality Assurance, From Theory to Implementation, 2004, str. 183.

Prednost metode velikega poka je manjša raba virov. Metoda, npr. za namen integracijskega testiranja, ne predvideva priprave modulov, ki posnemajo delovanje končanih programskih modulov. Poleg te prednosti ima manjše število testnih primerov.

Vendar sta obe prednosti izraženi le, ko gre za testiranje enostavnejše programske opreme (Galin, 2004, str. 186).

Metodi velikega poka komplementarna metoda je inkrementalno testiranje. Inkrementalno testiranje pomeni testiranje manjših programskih modulov, ki še ne predstavljajo celote ali niti ne celotnih modulov. K testiranju pristopimo bodisi »od spodaj-navzgor« (angl. *Bottom-up*) ali »od zgoraj-navzdol« (angl. *Top-down*). Prvi pristop pomeni, da najprej testiramo manjše module in jih brez znanih hroščev integriramo z drugimi moduli. Integrirani morajo usklajeno delovati in na tak način testiramo njihovo medsebojno delovanje in vpliv. V primeru pristopa »od zgoraj-navzdol« si testna ekipa pomaga z moduli, ki posnemajo delovanje končanih modulov. Inkrementalna metoda ima pred metodo velikega poka več prednosti, od katerih sta najbolj očitni:

- Testiranje se izvaja na relativno majhnih delih kode kot enotno ali integracijsko testiranje. Na ta način se detektira večji delež napak kot pri testiranju celotne programske rešitve.
- Popravljanje detektiranih napak je hitrejše ter enostavnejše, opravljeno je v zgodnejši razvojni fazi in zahteva manj virov.

Dejstvo, da se napake pri inkrementalni metodi odkrijejo ter odpravijo v zgodnejši fazi razvoja programske opreme, preprečuje, da bi se napake manifestirale v kasnejši fazi razvoja ali testiranja, ko jih je mnogo težje identificirati, razložiti in popraviti.

Ne glede na to, kateri pristop uberemo, je v obeh primerih testiranje potrebno izvajati kot zaporedje testnih primerov. Planiranje in dokumentiranje testnih primerov je torej pomembno iz vsaj štirih razlogov (Patton, 2006, str. 279):

- Organiziranost. Člani testne ekipe bodo dokumentirane testne primere učinkovito uporabljali za namen samega izvajanja testov kot tudi pri pregledovanju testnih rezultatov. Kakovostno dokumentirane teste je mogoče tudi hitro modificirati v nove primere.
- Ponovljivost. Skozi proces testiranja je potrebno določene teste izvajati večkrat, da bi se poiskalo nove hrošče ali potrdilo popravke prej identificiranih hroščev. Brez ustreznega dokumentiranja testnih primerov je nemogoče popolno ponoviti posamezne teste.
- Sledljivost. Pomeni poznati, kateri testi so bili izvedeni in kateri izpuščeni pri testiranju posamezne izdaje programske rešitve. Kateri testi so bili izvedeni s pozitivnim rezultatom in kateri neuspešno. Brez natančnega dokumentiranja testnih primerov je na omenjena vprašanja nemogoče odgovoriti.
- Dokazljivost. v nekaterih industrijskih panogah je potrebno dokazovati, da je ustrezno testiranje bilo izvedeno. Poročila o testiranju se morajo sklicevati na dokumentirane testne primere.

Poleg produkta je potrebno testirati tudi njemu pripadajočo dokumentacijo. Sestavljena je iz dokumentov, ki predstavljajo produkt, njegove lastnosti ter iz uporabniške dokumentacije, kot so navodila za uporabo, servisna dokumentacija ipd.

6.2.3 Predmet testiranja

V procesu testiranja potrebujemo predmet testiranja. Predmet testiranja je programska oprema oz. ena izmed njenih izdaj (angl. *release*). Med razvojem programske opreme nastane cela vrsta izdaj, ki se razlikujejo glede na čas in pogostost nastanka in glede na namen. Namen nastanka je lahko zgolj preverjanje integritete oz. prevedljivosti programske opreme ali pa je namen celovito, sistemsko preverjanje vseh funkcionalnosti. V začetnih fazah projekta nastanejo verzije, katerih namen je preverjanje pravilnosti zasnove produkta ter prototipne verzije. Kasneje nastanejo verzije, ki imajo polno funkcionalnost ter nato verzije, katerih namen je dvig kakovosti.

Nočna izdaja

Nočna izdaja (angl. *nightly build*) je izdaja programske opreme, ki se avtomatično zgradi preko noči oz. v času, ko se na kodi ne izvaja sprememb. Nočne izdaje se uporablja v fazi razvoja za potrebe testiranja prevedljivosti programske rešitve in vključevanja novih funkcionalnosti. V nočni izdaji je vedno najnovejša koda, odkriti manjši hrošči, pa so lahko popravljeni že v naslednji izdaji.

Alfa izdaja

Alfa izdaje (angl. *alpha release*) označujemo izdaje, na katerih se začnejo izvajati testi na celotni programski rešitvi. V tej fazi izvajamo teste prozorne škatle (angl. *clear* ali *white box test*), ki jim sledijo testi črne škatle (angl. *black box test*). Stanje programske kode v alfa fazi je navadno precej nestabilno in lahko povzroči sesutja programske opreme in izgubo podatkov. Za alfa izdajo je značilno, da naj bi bil obseg funkcionalnosti zaključen.

Beta izdaja

Beta izdaja (angl. *beta release*) je programska oprema, ki ima vse predvidene funkcionalnosti in je hkrati stabilizirana. Stopnja sledi alfa izdaji in njen glavni namen je dvig kakovosti. Fokus testiranja na beta izdaji je na uporabniški interakciji in uporabnosti programske opreme. Beta izdaje so lahko odprte ali zaprte za skupino uporabnikov. Uporabniki, ki testirajo to izdajo postanejo t. i. beta testerji. Zaprte beta izdaje so namenjene le izbrani skupini uporabnikov, medtem ko so odprte namenjene širši javnosti. Preizkuševalci prijavljajo hrošče, ki jih najdejo med testiranjem, in predlagajo manjše dodatne funkcionalnosti, ki bi bile smiselne v končni verziji programske opreme.

Končna izdaja

Angleški izraz *Release Candidate* je oznaka za izdajo, ki je potencialno končna izdaja programske opreme. V tej stopnji programska oprema stabilno deluje in je temeljito testirana. Vse funkcionalnosti so vključene in če v procesu končnega testiranja ni najdenih nobenih pomembnejših hroščev, postane izdaja končna, produkcijska izdaja (angl. *final release* ali *official release*).

Takšna verzija programske rešitve je primerna za uvedbo pri uporabniku. Ob tem so kasneje še vedno možni manjši ali urgentni popravki kode (angl. *patch*, *update*, *fix*) za odpravo kasneje odkritih nepravilnosti. Spreminjajo se lahko tudi dokumentacija in pomožne datoteke, ki sestavljajo programsko rešitev. Nove funkcionalnosti pa bodo dodane v prihodnjih planiranih izdajah.

6.2.4 Zaključevanje testiranja

Zaključevanje testiranja je tretja faza v procesu testiranja in pomeni predstavitev rezultatov ter izsledkov testiranja. Od vseh faz je najbolj ključna (Patton, 2006, str. 291), ker bo vse delo preteklih faz zaman in nepomembno, če se identificirani hrošči ne odpravijo.

Poročila za sledenje hroščem so v elektronski ali pisni obliki ali se pojavljajo kot izpolnjeni obrazci. Učinkovitejši pristop je podatkovna baza hroščev, ki omogoča celovitejše sledenje hroščem oz. njihovemu življenjskemu ciklu in preko metrik, ki jih ta orodja nudijo, nudi lažjo predstavo o tem, v kakšni fazi je projekt.

Predstavitev rezultatov je del zaključevanja testiranja in je v obliki testnih poročil različnih nivojev. Poročila nudijo informacijo, kaj je bilo opravljeno in kateri hrošči oz. napake v delovanju programske opreme so bile identificirane. Poročila pomenijo osnovno orodje za kontrolo kakovosti programske opreme.

6.3 Vrste testiranja

Testiranje črne škatle

Pri testiranju po načinu črne škatle (angl. *Black Box Testing*) je vsebina oz. implementacija testiranega modula testerju neznana. Znano mu je le, kaj naj modul dela oz. njegova funkcija, ki izhaja iz funkcijskih specifikacij modula (Naik & Tripathy, 2008, str. 163). Tester mora poznati predvidene odzive s katerimi se mora modul odzvati na znane vhodne parametre. Ni pa mu potrebno poznati algoritmov, ki privedejo do testiranih odzivov.

Pred samim testiranjem si mora, v fazi planiranja testiranja, tester pripraviti ustrezne testne primere. Njihova priprava sledi iz opisov modula, zahtev ter specifikacij modula. Gre za pripravo funkcionalnih testov. Dober tester bo previdno izbral vhodne parametre, s katerimi bo povzročil čim večje število napačnih odzivov in tako odkril čim večje število hroščev v implementaciji modula.

Testiranje prozorne škatle

Tovrstno testiranje (*angl. White Box Testing* ali *Clear Box Testing*) je testiranje internih struktur implementacije modula. Je nasprotno testiranju črne škatle, kjer gre predvsem za funkcionalno testiranje modula. Pristop k testiranju je bolj tehničen, za izvedbo testov ter samo pripravo so potrebna programerska znanja. Tester tudi v tem primeru uporabi različne vhodne parametre v modul, vendar tokrat z namenom, da bi testiral strukturo modula (Naik & Tripathy, 2008, str. 163). Testira različne poti izvajanja algoritmov modula, podatkovni tok, odločitve na osnovi vhodnih parametrov ipd. Testiranje prozorne škatle se običajno izvaja med testiranjem posameznih programskih modulov, torej na najnižjem nivoju testiranja. Možno pa ga je izvajati tudi na višjih nivojih, npr. med integracijo modulov v celoto.

6.4 Nivoji testiranja

Nekateri avtorji področja projektnega managementa zagovarjajo, da se testiranje izvaja šele po fazi izvedbe (Schwalbe, 2010, str. 313). Testiranje bi bilo izvedeno v skladu s testnim pristopom velikega poka (angl. *Big Bang*) ali ko bi razvoj tekel po slapovnem modelu. Vendar večina avtorjev identificira več pozitivnih dejavnikov drugačnega pristopa, ki je, da se testiranje izvaja skozi celoten proces razvoja programske opreme. S tega razloga ga opravljamo na več nivojih od testiranja posameznih enot, integracije medsebojno povezanih enot, vse do testiranja celotnega sistema.

Testiranje enot

Programsko opremo sestavlja več ločenih programskih enot oz. modulov. Testiranje enot (angl. *Unit Testing*) pomeni testiranje skladnosti modula z njegovimi specifikacijami. Testiranje modula običajno opravi razvijalec modula. Nastopi v vlogi testerja in opravi testiranje, tako da z različnimi vhodnimi parametri vzbuja odzive in jih primerja s predvidenimi rezultati. Testira tudi strukturo programskega modula, ki jo razvijalec najbolje pozna.

Testiranje opravimo ob koncu razvoja modula ali pa uporabimo angleško imenovan *Test Driven Development* pristop razvoja (Beck, 2003, str. 11), ki pomeni, da pred začetkom implementacije modula razvijemo testne primere, ki jih med izvedbo periodično izvajamo. Sprva so testi na testnih primerih neuspešni, kasneje, ko je razvite več kode, pa postanejo njihovi rezultati pozitivni.

Integracijsko testiranje

Integracijsko testiranje (angl. *Integration Testing*) je testiranje, kjer testiramo delovanje dveh ali več modulov skupaj kot povezano celoto. Namen integracijskega testiranja je identificirati morebitne napake pri medsebojni interakciji modulov. Časovno ga izvajamo po testiranju enot in pred sistemskim testiranjem. Pristopi k testiranju so lahko t. i. metoda

velikega poka ali pa pristop »od zgoraj-navzdol« ali »od spodaj-navzgor«. V prvem primeru ima prednost testiranje modulov, ki so arhitekturno višje. V drugem primeru velja obratno. V obeh primerih pa velja, da netestirane in nedokončane module nadomestijo moduli, ki posnemajo njihovo delovanje.

Pri integracijskem testiranju izvajamo teste t. i. črne škatle oz. tudi prozorne škatle. Izbira testov je odvisna od testnega osebja in njihove usposobljenosti.

Sistemsko testiranje

Sistemsko testiranje (angl. *System Testing*) je testiranje celotnega sistema oz. celotne programske rešitve. Testiranje izvajamo za integracijskim testiranjem. Namen sistemskega testiranja je preverjanje ali programska rešitev ustreza predhodno definiranim zahtevam.

Sistemsko testiranje običajno izvaja neodvisna testna ekipa. Pomembna je uporabna vrednost rešitve ter uporabniška izkušnja, zato se pri testiranju uporablja pristop črne škatle.

Prevzemno testiranje

Prevzemno testiranje (angl. *Acceptance Testing*) je testiranje celotnega sistema z namenom ali ga sprejeti oz. zavrniti v primerih, da so odkrite neustreznosti s poslovnimi ali uporabniškimi zahtevami. Prevzemno testiranje opravimo po sistemskem testiranju in je zadnje testiranje preden se sistem prične uporabljati s strani naročnika.

Testiranje opravijo člani posebne testne ekipe, ki je formirana znotraj podjetja, npr. iz članov projektne ekipe ali pa ga opravi ekipa, ki je neodvisno formirana zunaj podjetja. Prevzemno testiranje lahko opravijo tudi bodoči uporabniki sistema.

Regresijsko testiranje

Regresijsko testiranje je izločeno iz skupine, ki sestavlja opisane nivoje testiranja. Regresijsko testiranje je namreč mogoče oz. potrebno izvajati na vseh nivojih testiranja, kadar pride do sprememb modulov, ki sestavljajo sistem. Obseg regresijskega testiranja je odvisen od vpliva spremenjenega modula na celoten sistem.

7 ANALIZA IZBRANEGA PODJETJA

V poglavju je podan praktični del naloge, katerega splošen cilj je preučiti delovanje izbranega podjetja, predvsem njegovega razvojnega oddelka, identificirati njegove pomanjkljivosti in težave s katerimi se spopada, identificirati možne rešitve in jih uvesti v razvojni proces.

Obravnava projektov podjetja je omejena in je opravljena predvsem z vidika managementa časa in učinkovitosti dela razvojnega oddelka ter managementa kakovosti. V nekaterih

primerih obravnavam podjetje nekoliko širše, npr. tudi z organizacijskega vidika, ko je leta povezan z omenjenima področjema.

Čas opazovanja delovanja podjetja in managementa projektov razvoja je nekaj več kot 12 mesecev pred izvedenimi spremembami in nadaljnjih 12 mesecev od začetka sprememb. V tem času je bilo izvedenih 7 projektov, od tega 2 za podjetje relativno velika, s trajanjem približno 10 mesecev. Ostali so bili manjši, ki so trajali od 3 do 4 mesece.

Poglavje je razdeljeno na:

- predstavitev podjetja in njegove organizacije, produktov ter projektov,
- opis modela CMMI ter CMMI-DEV,
- identifikacijo pomanjkljivosti razvoja programske opreme glede na model CMMI,
- identifikacijo možnih rešitev, možno odpravo pomanjkljivosti ter analizo vpeljanih sprememb z modelom CMMI,
- opis modela TMMI in analizo testnega procesa po modelu TMMI ter
- možne in potrebne nadaljnje izboljšave.

V prvem delu poglavja je opisano obravnavano podjetje. Opisane so splošne ter organizacijske lastnosti podjetja, predvsem tiste, ki neposredno vplivajo na izvajanje procesov. Opisano je izvajanje projektnega managementa, ki je v tem delu omejeno na opis pred uveljavljenimi spremembami. Opisana je panoga delovanja podjetja ter produkti, s katerimi nastopa na trgu. Na koncu tega dela je na kratko opisan razvojni oddelek.

V drugem delu poglavja opišem model CMMI ter specifičen model CMMI-DEV, ki se uporablja za razvojno usmerjena podjetja.

V tretjem in četrtem delu analiziram delovanje podjetja ter identificiram težave, s katerimi se spopada. Podjetje primerjam z zrelostnim modelom CMMI, ocenim, v katero stopnjo spada, in identificiram potrebne spremembe, da bi podjetje pridobilo višjo stopnjo. Sledijo opisi predlogov rešitev za identificirane težave, potrebni ukrepi ter njihova vpeljava v proces. Podana je analiza učinkov vpeljanih sprememb, ki temelji na opazovanju delovanja razvojnega oddelka po vpeljanih spremembah. Analiziram še morebitno pridobitev stopnje po modelu CMMI.

V posebnem delu so opisani ukrepi za izboljšanje kakovosti programske opreme. Ukrepi so bolj tehnične narave in so pomembna pridobitev glede na preteklo stanje.

Analiziram tudi testne aktivnosti podjetja, ter podobno kot za model CMMI, umestim podjetje v model TMMI. Identificiram, kaj botruje umestitvi v izbrano stopnjo ter predlagam potrebne ukrepe, da bi podjetje pridobilo stopnjo tudi v modelu TMMI.

V zadnjem delu poglavja podam predloge za nadaljnje delo oz. katere spremembe je po mojem mnenju v prihodnje še potrebno vpeljati v razvojni proces podjetja. Potrebno se je

namreč zavedati, da je posamezen proces mogoče vedno dodatno optimizirati, ga narediti bolj učinkovitega in posledično izboljšati produkt. Na to dejstvo nas spomnijo težave, ki so seveda še vedno prisotne, in stanja ni smotrno jemati takega kot je, ampak je potrebno neprestano iskati rešitve za preostale težave.

7.1 Predstavitev podjetja

Obravnavano podjetje je razvojno usmerjeno in ima svoj razvojni oddelek. Oddelek deluje 5 let in v tem času je razvil lastno družino naprav, ki se uporablja v navigacijskih aplikacijah namenjenih sledenju v kopenskem transportu ter v navtiki. Naprava je med uporabo nameščena na vozilo oz. plovilo in z uporabo satelitskih ter zemeljskih komunikacijskih omrežij omogoča spremljanje pozicije vozila v realnem času. Razvita družina naprav je v skladu z najnovejšimi tehnološkimi smernicami. Podobne konkurenčne naprave, ki nastopajo na trgu, v mnogih lastnostih presega.

Podjetje je razvijalec strojne opreme oz. ima razvit lasten hardver in je hkrati razvijalec vgrajene (angl. *embedded*) programske opreme, s katero podpre delovanje strojne opreme.

Strojna oprema naprave ima vgrajen 32-bitni mikrokrmilnik, ki zagotavlja delovanje cele vrste navigacijskih ter komunikacijskih aplikacij, od sledenja in poročanja o poziciji, do prenosa diagnostičnih podatkov ter datotek. Aplikacije so realizirane v programski opremi, ki teče v mikrokrmilniku. Programska oprema je neločljiva, posebej prilagojena napravi in vsebuje gonilnike, namenjene podpori delovanja komunikacijskih ter navigacijskih modulov in drugih perifernih sklopov, senzorjev, merilnikov, ki so del strojne opreme.

Podjetje deluje na globalnem trgu. Za naročila se udejstvuje na javnih mednarodnih razpisih in vzdržuje tudi svojo distribucijsko mrežo, preko katere trži naprave, za končne uporabnike.

Naročila pridobljena na razpisih oz. te naprave so namenjene in prilagojene vnaprej znanim kupcem. Prilagodi ali na novo se razvije tako strojna oprema kot tudi programska oprema. Predvsem razvoj novih funkcionalnosti, s katerimi se zadovolji razpisnim pogojem, zahteva največ časa ter drugih virov. Podjetje se razvoja novih izdelkov ter prilagoditve obstoječih rešitev loti s projektnim načinom dela.

Podjetje je v preteklih letih pridobilo nekaj mednarodnih naročil, pretežno državna naročila iz držav članic evropske unije ali držav s področja Balkana. Trajanje tovrstnih projektov je od 4 do 12 mesecev, pri čemer je njihova ocenjena velikost od 12 do 60 človek-mesecev. Da bi produkt v celoti ustrezal razpisnim pogojem so običajno potrebne modifikacije strojne opreme, več časa ter virov pa je potrebno vložiti v razvoj modifikacij programske opreme.

Poleg projektov za mednarodna naročila tečejo vzporedno tudi projekti razvoja novih produktov, ki so za čas, ko tečejo projekti z razpisov, prekinjeni oz. dobijo nižjo prioriteto.

Vzporedno torej podjetje razvija sorodne produkte, ki se od osnovnih različic razlikujejo po številu vgrajenih funkcionalnosti ali pa so bistveno drugačni in namenjeni drugim trgom in povsem drugim namenom uporabe.

Organizacijska piramida podjetja je plitka. Podjetje je vodeno s strani lastnika ter glavnega direktorja v eni osebi. Organizirano je na tradicionalne funkcijske enote, nabavo, prodajo, marketing, proizvodnjo, skupne službe in ima močno zastopan razvojni oddelek.

V podjetju je zaposlenih od 15 do 20 inženirjev tehničnih strok. Pretežno gre za kadre z izobrazbo s področja elektrotehnike ali informacijske tehnologije. Njihovo natančno število je odvisno od trenutnih potreb po človeških virih. Zaposleni so visoko izobraženi, so izkušeni in usposobljeni za opravljanje kompleksnih razvojnih nalog.

Predstavitev razvojnega oddelka

Razvojni oddelek se deli na 3 pod-oddelke, pod-oddelek za razvoj strojne opreme, pod-oddelek za razvoj vgrajene programske opreme oz. firmvera ter pod-oddelek za razvoj aplikativne programske opreme.

Posamezni pod-oddelki imajo od 3 do 7 članov, odvisno od trenutnih potreb oz. glede na velikost projekta, ki se izvaja. Vsak pod-oddelek ima svojega vodjo oz. vsaj enega vodilnega inženirja, ki je neposredno odgovoren projektnemu managerju, v praksi pa neposredno tudi direktorju oz. lastniku podjetja.

Delo na nalogah je pretežno timsko. Enostavnejše in krajše naloge se izvajajo samostojno, a usklajeno z drugimi člani ekipe.

Uporabljan programski jezik ekipe za razvoj vgrajene programske opreme je jezik C.

Vsi pod-oddelki so ustrezno opremljeni glede merilne ter infrastrukturne opreme. Na voljo imajo dovolj kakovostne opreme, da ta ne povzroča zamud pri izvajanju nalog.

Projektni managerji niso del razvojne ekipe, temveč se ji začasno pridružijo za izvedbo projekta pridobljenega na razpisu. Oseba, ki je pridobila naročilo, je odgovorna za izvedbo naročila in prevzame vlogo projektnega managerja. Management tehnične plati projekta prevzame eden izmed vodilnih inženirjev.

Za management projektov popolnoma novih produktov, ki tečejo vzporedno omenjenim projektom, ter za manjše interne projekte, vlogo projektnega managerja prevzame vodilni inženir razvojne ekipe.

7.2 Model CMMI

Da bi praktični del naloge izhajal iz teorije najboljših praks sem preučil teorijo ocenjevanja zrelosti podjetij, ki uporabljajo v svojem delovanju projektni management. Uporabil sem

zrelostni model CMMI, ki je uporabljan kot pomoč organizacijam pri izboljšanju procesov (Software Engineering Institute, 2010 v nadaljevanju SEI).

Postopek uporabe modela je analiza trenutnega stanja delovanja podjetja ali njegove manjše enote, določitev trenutne stopnje in določitev potrebnih prihodnjih korakov, da bi se pridobilo stopnjo na lestvici modela. Model ima 5 zrelostnih stopenj. Kratek opis posameznih stopenj modela CMMI je (Peters, 2008, str. 51):

- Stopnja 1: Pristop managementa k projektom je nekonsistenten in se razlikuje od projekta do projekta. Razmere lahko opišemo z izrazom kaotične. Večina časa je posvečena t. i. gašenju požarov in le malo časa se posveča preprečevanju nastanka težav.
- Stopnja 2: Situacija je manj kaotična kot v prejšnji stopnji. Uveljavlja se projektno razmišljanje. Nekaj časa se posveti analizi zahtev, managementu časa, stroškov in zagotavljanju kakovosti. Uporabi se izkušnje s preteklih projektov.
- Stopnja 3: Projektno razmišljanje se uveljavi na nivoju podjetja. Projektni manager je postavljen in ima podporo s strani vodstva podjetja. Izvedba projekta je koordinirana tudi v povezavi z drugimi oddelki podjetja, npr. marketingom. Uveljavljeni so standardi in uporabljeni na aktivnostih projekta. Procesi so dobro dokumentirani, dokumentacija pregledana in odobrena. Npr. testni načrt je odobren pred izvedbo testiranja.
- Stopnja 4: Management projekta podjetju ne predstavlja težav. Fokus je na doseganju kakovosti in zanesljivosti programske opreme. Management kakovosti je obvladana disciplina managementa projektov.
- Stopnja 5: Najvišja stopnja. Poteka stalno izboljševanje stopnje 4. Poskuša se privzemanje novih tehnologij in optimizacija procesov. Rezultati sprememb so merljivi in uporabljeni za potrjevanje ustreznosti sprememb in preprečevanje napak.

Model CMMI-DEV

Področju razvoja novih produktov je namenjen prirejen model CMMI-DEV (angl. *CMMI for Development*). CMMI-DEV ponuja najboljše prakse kot vodilo razvojno usmerjenim podjetjem, da bi le-ta ponudila strankam in končnim uporabnikom kar najkakovostnejše produkte (SEI, 2010).

CMMI-DEV vsebuje 22 procesnih področij, od tega je 16 splošnih in jih najdemo v vseh CMMI modelih, enega najdemo v več modelih, 5 področij pa je specifičnih CMMI-DEV modelu.

Med splošnimi področji modela najdemo področja povezana z organizacijskimi lastnostmi organizacije, s projektnim managementom, zagotavljanjem kakovosti ter drugimi.

Za pridobitev stopnje mora podjetje:

- razumeti evolucijsko lestvico modela,
- opraviti proces presoje in se uvrstiti ali uvrstiti svojo enoto v identificirano stopnjo,
- identificirati cilje, ki jih mora doseči. Za dosego ciljev, npr. ustanoviti procesno skupino, ki bo skrbela za dosego ciljev. V primeru prehoda iz stopnje 1 na stopnjo 2, bo skupina zagotovila bolj disciplinirane procese še preden bo stopnja 2 dosežena,
- doseči vse cilje, preden ponovno presodi in se potencialno uvrsti ali uvrsti svojo enoto v višjo stopnjo.

7.3 Identifikacija pomanjkljivosti

V 12-ih mesecih pred uvedbo sprememb je v podjetju potekalo več projektov, ki so bili izvedeni po uspešno pridobljenih mednarodnih razpisih. Njihov rezultat so bile, v skladu z razpisnimi pogoji, razvite naprave.

Podjetje je projekte končalo in se med njihovim izvajanjem spopadalo s tipičnimi težavami, ki jih glede na preučeno literaturo ter v skladu z lastnimi izkušnjami na projektih razvoja programske opreme, pričakujem. Čeprav so bili po obsegu ter vsebini projekti različni, identificiram podoben vzorec dogodkov in težav, pretežno povezanih s časovno in kakovostno komponento projektov.

7.3.1 CMMI analiza

Z analizo oz. primerjavo med dogajanjem na projektih ter opisom posamezne stopnje (SEI, 2010, str. 27) sem za omenjeno obdobje identificiral stopnjo zrelosti podjetja, ki je stopnja 1. Poleg splošnega opisa sem preučil tudi posamezna procesna področja in katerim kriterijem mora podjetje zadostiti, da bi imelo možnost uvrstitve v stopnjo 2.

Procesna področja, ki jim mora podjetje ali njegova enota zadostiti za pridobitev stopnje 2, so:

- Planiranje (angl. *Project Planning*),
- Spremljanje in kontrola (angl. *Monitoring and Control*),
- Zagotavljanje kakovosti produktov in procesa (angl. *Product and Process Quality Assurance*),
- Management odnosov z dobavitelji (angl. *Supplier Agreement Management*),
- Management konfiguracij (angl. *Configuration Management*),
- Merjenje in analiza (angl. *Measurement and Analisys*).

Procesno področje planiranja predvideva disciplinirano izvajanje aktivnosti, kot so določanje obsega projekta, izdelava strukture WBS, določitev faz življenjskega cikla projekta, določanje atributov aktivnostim, določanje potrebnih virov, izdelava časovnice projekta, zagotavljanje potrebnih znanj in tehnologij ipd.

Procesno področje spremljanje in kontrola ima namen zagotoviti razumevanje o poteku oz. napredku projekta, da se bodo lahko vršili korektivni ukrepi, ko bo napredek bistveno odstopal od načrta. Področje predvideva izvajanje aktivnosti, kot so primerjanje planiranih atributov z dejanskimi, spremljanje časovnice, spremljanje doseganja mejnikov, izvajanje korektivnih ukrepov ob identificiranih težavah ipd.

Procesno področje zagotavljanja kakovosti produktov in procesov predvideva uporabo standardov za namen primerjave načina izvajanja procesov z načinom izvajanja, ki ga predvideva standard. Na podoben način predvideva uporabo standardov tudi na samih produktih ter zahteva aktivno reševanje težav ob zaznanih neskladnostih med standardi in dejanskim stanjem produktov.

Ocenjujem, da so razlogi za umestitev podjetja v stopnjo 1:

- Pri opazovanju projektov razvoja programske opreme je težko identificirati vse faze projektnega managementa. Najlažje je identificirati fazo izvedbe, medtem ko so faze, kot so zagon, planiranje ter zaključek, opravljene površno. Procesi niso definirani in so nezaključeni. V času kriz se opušča nekatere aktivnosti.
- Identificirati je možno nepravočasno in neustrezno odločanje, ki ne temelji na realnih dejstvih. Tehnike za krajšanje časovnice se med izvedbo ne uporabljajo.
- Pristop k managementu projektov je nekonsistenten. Podjetje ne zagotavlja ustaljenega poslovnega okolja, zato sta učinkovitost managementa projektov in izid projektov odvisna od posameznikov. Obravnavano podjetje nima specialistov projektnih managerjev, temveč je management posameznih projektov zaupan posamezniku, ki je na delovno mesto delegiran začasno, za trajanje projekta.
- Podjetje ne planira kakovosti in nima ustaljenih mehanizmov za zagotavljanje kakovosti. Podjetje ne uporablja standardov kakovosti s področja razvoja programske opreme. Procesi kontrole kakovosti so omejeni na testiranje, ki je neučinkovito in deluje po neustreznem pristopu. Odkrite napake v delovanju programske opreme so neustrezno dokumentirane in neustrezno sporočene odgovornim.
- Produktne zahteve niso ustrezno zajete in revidirane. Zajem zahtev se ne odrazi v projektnem načrtu. Ustreznega upravljanja s spremembami zahtev ni, in spremembe se ne odrazijo v prilagojenih projektnih načrtih. Zajem zahtev naročnikov je večkrat napačen ali narobe razumljen.

Uvrstitev podobnih podjetij v posamezne skupine bi pokazalo, da je velika večina podjetij na zrelostni stopnji 1. Nekaj manj jih je doseglo stopnjo 2. Število podjetij, ki so dosegla višjo stopnjo, nato eksponentno upada. Le nekaj podjetij s področja razvoja programske opreme se lahko pohvali s stopnjo 5 (Peters, 2008, str. 52).

Teme in pomanjkljivosti, ki so tesneje povezane z obravnavano temo dela sem natančneje preučil. Ocenjujem, da omenjene pomanjkljivosti vodijo do produktov slabše kakovosti ter prispevajo k neučinkoviti izrabi časa med razvojem produktov.

Planiranje

V literaturi ter v zaključnih nalogah je za manjša podjetja večkrat opisana naslednja situacija.

Faza planiranja je v manjših podjetjih, ko ob začetku projektov prevladuje optimizem ter občutek, da bo že nekako šlo, nemalokrat zanemarjena faza projektnega managementa. Aktivnosti so definirane le v glavah deležnikov projekta in niso formalizirane v obliki dokumentov. Trajanje posameznih aktivnosti je ocenjeno na pamet, brez dokumentirane uporabe predhodnih izkušenj ter dognanj s preteklih sorodnih projektov.

V obravnavanem podjetju pred prenovo procesov identificiram opisano situacijo. Projektno delo je sekundarno delo vodilnemu inženirju. Iz tega razloga je aktivnosti planiranja premalo in planiranje je opravljeno površno ter pomanjkljivo.

Aktivnosti, ki jih ni ali so površno opravljene, so:

- Definicija aktivnosti, v kateri se ne identificira vseh aktivnosti, ki jih je potrebno na projektih izvesti. Identifikacija aktivnosti poteka »od zgoraj-navzdol«, vendar ni dovolj podrobna.
- Razvrščanje aktivnosti je iz zgoraj navedenega razloga narejeno površno. Mrežni diagrami se ne naredijo.
- Za identificirane aktivnosti se opravi ocena potrebnih virov, vendar se ne upošteva časovne razpoložljivosti.
- Za identificirane aktivnosti se opravi ocenjevanje njihovega trajanja na osnovi ekspertnih presoj. Uporablja se izkušnje preteklih projektov. Ocenjujem, da so ocene sprva ustrezne, vendar ne upoštevajo razpoložljivosti virov oz. ne upoštevajo koledarja razpoložljivosti, zato se kasneje izkažejo kot preveč optimistične.
- Časovnica projekta je narejena površno in ne vsebuje vseh aktivnosti. Sestoji iz datuma začetka in konca projekta ter glavnih mejnikov. Posamezni inženirji naredijo podrobnejše časovnice za večje sklope del, kot so, npr. razvoj novih večjih programskih modulov ali za testiranje.

Izvedba

O zamudah pri izvajanju projektov, kjer se aktivnosti časovno ne planira, lahko govorimo le pogojno. Česar ne planiramo, ne moremo meriti in se odzvati s pravilnimi odločitvami. Kljub temu trenutek, ko mora biti produkt končan in izdelan v predvideni seriji, pride in če projekt ni zaključen, objektivno govorimo o zamudi projekta.

Posledice neplaniranja so časovne stiske in stresne situacije, ki negativno vplivajo na učinkovitost izvedbe ter brezhibnost dela razvojne ekipe. Pojavijo se nepotrebne napake v programski kodi. Te je toliko težje odkriti, ker so prva žrtev časovne stiske, aktivnosti za zagotavljanje ter kontrolo kakovosti. Kasneje, ko je napaka odkrita je relativno enostavno

identificirati vzrok za njen nastanek ali vzrok, da ni bila odkrita preden je bila naprava poslana stranki. Vzrok identificiram v neizvedenem testiranju.

Ugotavljam, da je odsotnost managementa časa v fazi izvedbe pogosto krivec za izdelke slabe kakovosti, ki imajo več odpovedi, delujejo nepravilno ter imajo manj želenih funkcionalnosti. Ko taki izdelki pridejo do strank in so v uporabi, lahko bi rekli masovno testirani, je pričakovati večje število zahtev po zamenjavah ter vzdrževanju.

Kakovost

Podjetje se srečuje s težavami zaradi produktov slabe kakovosti, ki se kaže v velikem številu odpovedi ali neustreznem delovanju naprav.

Trdim, da težave izhajajo iz splošnega nerazumevanja pomena kakovosti in neplaniranja časa namenjenega aktivnostim za zagotavljanje ter kontrolo kakovosti.

Člani razvojne ekipe so mnenja, da je nekakovost posledica stalne bitke s časom in lovljenja rokov za odpremo produktov k stranki. Tudi v primeru projektov iz preteklosti je osnovno orodje kontrole kakovosti, tj. testiranje, zanemarjeno in se za aktivnosti testiranja že med grobim planiranjem ni rezerviralo časa.

7.3.2 Model razvojnega procesa

Po analizi izvajanja razvojnega dela ugotavljam, da izbrano podjetje nima formalno ali kako drugače izbranega modela razvoja programske opreme. Po preučevanju cele vrste modelov s področja, od formalističnih do modernejših, agilnih modelov, zaključujem, da je način dela še najbolj podoben modelu »Kodiraj in popravljaj«, ki je opisan v poglavju 5.3. Sklepam, da je izbira modela pričakovana za podjetja, ki prepuščajo dogodkom svojo pot in ne planirajo. Patton (2006, str. 33) opisuje model kot intuitivno izbiro, ko razvojna ekipa ne poskuša zavedno izbrati drugega modela.

S teorijo modela »Kodiraj in popravljaj« se ujema tudi potek testiranja izdaj programske opreme. Nova izdaja je pogosto na voljo ter vgrajena v produkt preden je končano testiranje prejšnje izdaje. Ocenjujem, da je posledica delovanja po identificiranem modelu tudi veliko ponovnega dela na programskih modulih.

7.4 Ukrepi za odpravo pomanjkljivosti

Da bi podjetje izboljšalo učinkovitost razvojnega oddelka in imelo možnost doseči višjo stopnjo po modelu CMMI sem identificiral možne ukrepe za odpravo poglavitnih pomanjkljivosti za pridobitev stopnje 2. Iz analize nabora CMMI področij, ki so našteta v poglavju 7.3 (SEI, 2010, str. 33) in so potrebna za stopnjo 2, ter prej identificiranih pomanjkljivosti sem določil področja in ukrepe na katere naj se podjetje osredotoči.

Glede na pričakovane učinke naj se podjetje sprva osredotoči na uvedbo naslednjih sprememb:

- Potrebno je izobraziti specialiste, projektne managerje oz. pridobiti potrebna managerska znanja. Posameznikom je potrebno omogočiti usmerjeno izobraževanje s področja projektnega managementa.
- Planiranju je potrebno posvetiti več časa in virov. Predvsem na področju planiranja oz. managementu časa na splošno je potrebno odpraviti pomanjkljivosti, ki so nepopoln pregled nad celotnim obsegom dela oz. nepopolna identifikacija vseh aktivnosti projekta. Potrebna je uporaba računalniških orodij za vodenje projektov, uporaba planiranja potrebnih virov, uporaba orodij, npr. Gantt diagrama, tehnik ocenjevanja trajanja aktivnosti ipd.
- V fazi izvedbe identificiram potrebo po aktivnejši kontroli izrabe časa. Uporabo časovnice za pravočasno odločanje o uporabi tehnik skrajševanja časovnice, ki bodo projekt peljale po planirani poti. Iz modela »Kodiraj in popravljaj« je potrebno preiti v drugačen, bolje definiran model razvoja programske opreme. Predlagam uporabo agilnega pristopa, ki ga kot možnega predvideva tudi CMMI.
- Potrebno je dvigniti zavest o pomenu kakovosti, ki naj se odrazi v večji količini aktivnosti za zagotavljanje in kontrolo kakovosti. Sprva je potrebno spremeniti pristop k testiranju, ki mora biti bolje definirano in podprto z izdelavo testne dokumentacije. Potrebna je uporaba standardov kakovosti, predvsem s področja razvoja programske opreme.
- Potrebno je natančneje definirati zahtevane lastnosti produkta že v zgodnji fazi projekta. Lastnosti produkta se morajo odraziti v projektnem načrtu in posledično v aktivnostih v fazi izvedbe. Zajete zahteve je potrebno ustrezno dokumentirati. Možna je uporaba standardov, npr. IEEE 1016, ki opredeljuje vidike kakovosti opisov specifikacij programske opreme.

Uvajanje sprememb je potekalo v korakih skozi zadnjih 12 mesecev. V tem času je bil izveden en večji projekt ter izvedeni trije manjši. Največ sprememb je uvedenih v fazo planiranja, v fazo izvedbe ter spremljanja in kontrole. Rezultati bolje opravljenega dela se kažejo v izdelanih diagramih poteka projektov, mrežnih diagramih, Gantt diagramih ter kasneje med izvedbo v lažji kontroli izvedbe ter hitrejšem in hkrati ustreznejšem ukrepanju.

7.4.1 CMMI analiza sprememb

Primerjava podobnih projektov pred in po uvedenih spremembah jasno kaže, da so projekti izvedeni nekoliko hitreje, ne bistveno. Vendar so produkti, ki so rezultat projektov kakovostnejši, kar se kaže predvsem v zanesljivejšem delovanju naprav in predvsem v manj vzdrževalnih aktivnostih po navideznem koncu projektov.

Cilj uvedenih sprememb je bila opredelitev ali je podjetje s spremembami pridobilo CMMI stopnjo. Primerjava opisov CMMI stopnje 1 ter stopnje 2 in analiza ali so razlogi za uvrstitev podjetja v stopnjo 1 še prisotni da naslednje rezultate:

- Faze projektnega managementa je možno lažje identificirati. Predvsem faza planiranja je dobro definirana in njeni rezultati so jasno vidni. Faza izvedbe sledi planiranju. Med izvedbo se izvaja ukrepe, ki so potrebni, da projekt ne zamuja. Fazo zaključevanja je težje identificirati. Prav tako nisem zadovoljen z dejstvom, da se v času kriz poskuša nekatere aktivnosti, npr. tiste povezane s kakovostjo, še vedno preskočiti.
- Ukrepi v fazi izvedbe ter spremljanja in kontrole so zaradi boljšega pretoka informacij izvedeni pravočasno oz. prej kot v preteklosti. Občasno se uporablja tehnike za krajšanje časovnice.
- Podjetje še nima specialistov projektnih managerjev. Vendar posamezniki, ki opravljajo to vlogo, več časa namenijo managementu projektov. Zavedanje o pomenu projektnega dela je višje.
- Zavedanje o pomenu kakovosti je višje. H kakovosti se pristopi predvsem iz tehničnega zornega kota, ki pomeni, da je cilj izdelati napravo z manj napakami ter zanesljivejšo. Cilj je bil dosežen z inženirskimi ukrepi, izobraževanjem zaposlenih ter povečanjem aktivnosti kontrole kakovosti, predvsem s celovitejšim testiranjem. Podjetje tudi vztraja pri vzdrževanju visoke kakovosti z uporabniškega vidika. Naprave so opremljene z velikim številom funkcionalnosti, ki omogočajo uporabnikom celo vrsto aplikacij.
- Zajem zahtev je ustreznejši ter dokumentiran. Zahteve se odrazijo v projektnem načrtu. Kasnejši management sprememb pa bi bilo potrebno bolj natančno izvajati in predvsem analizirati posledice sprememb na izvedbo. Identificiram, da je največji problem širitev obsega del, po kateri se ne izvede analize, kakšne posledice bo sprememba imela na izvedbo in končanje projekta.

S kontinuiranim projektnim načinom dela je ekipa dvignila nivo učinkovitosti. Delo na projektih je bolj rutinirano in rezultati projektov odvisni od delovanja celotne ekipe in ne le izjemnega prizadevanja njenih posameznih članov. Razvojna ekipa se z manj tveganji loti kompleksnejših ter obsežnejših projektov. Rezultati so kakovostnejši, manj je vzdrževalnih del na preteklih projektih.

Ugotavljam, da podjetje po spremembah lahko pogojno uvrstimo v višjo stopnjo, če se omejimo le na splošen opis stopnje. V primeru, da se osredotočimo na posamezne podrobnosti, ki so navedene v CMMI-DEV dokumentu, pa bo potrebno pomanjkljivosti še odpraviti in se osredotočiti še na nekatera področja.

Planiranje

Čas rezerviran za planiranje se je podaljšal za faktor 2 ali celo 3, ki ga ocenjujem empirično ter deloma na podlagi v informacijski sistem podjetja zabeleženih vrst dela, pred in po prenovi. Ocena je kljub temu zelo približna.

Ker je bilo prej planiranje izvedeno površno, se za planiranje porabi znatno več časa ter uporablja računalniška orodja namenjena managementu projektov. Poudarek je na planiranju časa, kjer se:

- identificira vse potrebne aktivnosti. Identifikacija se vrši na osnovi specifikacij bodoče naprave in jo vršijo vsi člani projektne skupine na posebej sklicanih sestankih. Identificirane aktivnosti se vnese v WBS strukturo. Identificira se kritične aktivnosti.
- preuči se koledar virov in uskladi aktivnosti z drugimi projekti, ki tečejo vzporedno, da ne bi prihajalo do zamud zaradi nerazpoložljivosti virov.
- aktivnosti se razvršča, glede na identificirane povezave in glede na izkušnje optimizira vrstni red izvajanja aktivnosti.
- iz pridobljenih podatkov se izdela časovnico projekta, ki je osnova za kontrolo izvedbe projekta.

S povečanjem aktivnosti v fazi planiranja, je dosežen ustrezen prihranek časa med izvedbo ter hkrati dvig kakovosti produktov do trenutka odpreme k naročniku ter manj vzdrževalnih del po koncu projekta.

Izvedba

Med izvedbo se aktivno spremlja napredek, primerja planirano ter doseženo ter na podlagi merljivih dejstev, pravočasno ukrepa, torej nadzoruje časovnico.

Možna je uporaba tehnik krajšanja časovnice oz. pravočasno odločanje o ukrepih, za zmanjšanje časovnih zaostankov, najpogosteje s prerazporeditvijo članov razvojne ekipe na dela na kritični poti projekta.

V primeru, da prerazporeditev virov ni možna, npr. ker viri niso razpoložljivi se delo na projektnih z nižjo prioriteto začasno ustavi.

Ukrepi so dogovorjeni na rednih sestankih razvojne ekipe ter usklajeni z vodstvom podjetja. Redni sestanki razvoja so tudi mesto za podajanje predlogov in izražanje težav, ki nastajajo med razvojem. Spremembe, ki so posledica odstopanja od načrta, se sproti vnaša v časovnico projekta ter analizira možne posledice na prihodnje izvajanje nalog.

Večja sprememba v fazi izvedbe je spremenjen model razvoja programske opreme.

7.4.2 Uvedba agilnega modela razvoja programske opreme

Med opazovanjem delovanja razvojne ekipe sem identificiral, da ekipa ne uporablja formalno določenega razvojnega modela. Delovanje je bilo najbolj podobno modelu »Kodiraj in popravljaj«, ki ima nekaj zelo primernih lastnosti, npr. zahteva malo formalnosti, vendar v obravnavanem primeru ni optimalen.

Odločitev, da ekipa izbere in uveljavi drugačen model, je bila potrebna z vidika rabe časa in dviga kakovosti. Ker smo nekateri člani razvojne ekipe že imeli predhodne izkušnje z različnimi razvojnimi modeli, je bil predlagan in vpeljan agilni model razvoja programske opreme (angl. *Agile development model*).

Uvedba agilnih metod razvojni ekipi omogoči bolj sistematičen način dela, ki prispeva k bolj predvidljivim rezultatom dela, obenem pa tudi drugi oddelki z večjo zanesljivostjo pričakujejo, kdaj bodo naloge opravljene.

Agilne metode predvidevajo delitev časa v intervale z enakim trajanjem (angl. *sprint*). Planiranje za posamezen interval oddelek razvoja ter deležnike projektov ter produktne managerje, prisili narediti seznam nalog za več tednov vnaprej. Tem nalogam prirediti prioriteto, medsebojne odvisnosti ter trajanje.

Ocenjevanje trajanja posameznih aktivnosti projekta projektni manager izvaja skupaj z drugimi deležniki projekta. Gre za ocenjevanje po ekspertni presoji oz. analogno. Posebej vidno vlogo imajo tudi dejanski izvajalci posameznih aktivnosti, ki morajo izpolniti zaveze glede časovnih rokov in se jim zaradi soustvarjanja ocen intenzivneje zavežejo.

Po določitvi nalog ter razporejanju kadrov na posamezne naloge, se razvojniki zavežejo k izvajanju teh nalog. Predvsem zaveza k izpolnjevanju ciljev nalog in dejstvo, da so cilji jasno določeni, omogoči osredotočenost na naloge ter obenem v dobršni meri preprečuje, da se fokus iz prioritetnih nalog prestavi na priložnostne naloge, ki so prej bistveno skrajševale čas rezerviran prioritetnim nalogam.

Kot najprimernejše trajanje posameznega časovnega intervala sta se v nekaj iteracijah pokazala dva delovna tedna.

Po koncu posameznih časovnih intervalov oz. pred začetkom novih se ekipa zbere na rednem sestanku (angl. *sprint review*), ki ima več namenov:

- Pregledati ter analizirati opravljeno delo v preteklem intervalu ter ga primerjati s planiranim na predhodnem sestanku.
- Identificirati tudi nepredvideno delo, ki ga je bilo potrebno urgentno opraviti in analizirati, kako je to delo vplivalo na planirane naloge in kakšni so vzroki za pojav nepredvidenega dela. Odgovoriti tudi na vprašanja, ali so vzroki za izvajanje

nepredvidenega dela opravičljivi ter uvesti ukrepe, da bi se v prihodnosti nepredvideno delo ne pojavilo.

- Pomemben del sestanka je planirati delo za vnaprej. Določiti, katere naloge iz znanega nabora se bodo izvajale v nekaj naslednjih intervalih, jim določiti prioritete ter določiti izvajalce. Posameznemu članu ekipe se v naslednji interval planira obremenitev, ki je realno izvedljiva. Pri tem je vodilo, da se sposobnosti razvijalca, čimbolj realno oceni.

Na sestanku identificiramo tudi popolnoma nove naloge, jih umestimo v širši okvir posameznega projekta in v grobem določimo prioriteto ter termin, v katerem bi se naloga predvidoma izvedla. Diskusija o novih nalogah je koristna tudi zaradi obveščanja članov ekipe o pomenu nalog za posamezen projekt, razumevanju vsebine ter ciljev nalog.

Pomemben del sestanka je tudi predstavitev rezultatov končanih nalog drugim članom oddelka oz. zaposlenim, ki so na tak ali drugačen način povezani z nalogo, bodisi kot uporabnik ali kot notranji naročnik.

V vmesnem času med sestanki poteka izvedba nalog. Stanja nalog se vodi na tabli (angl. *scrum board*), ki se v fizični obliki nahaja v prostoru, kjer se izvaja razvojno delo. Tabla predstavlja stanje nalog za trenutni interval ter nabor nalog (angl. *backlog*) za nekaj intervalov vnaprej. Obstaja tudi preslikava table v elektronski obliki v spletno aplikacijo. Namen podvajanja je v tem, da je tabla v fizični obliki ves čas na voljo in na vpogled vsem članom razvojnih ekip. Hkrati pa tudi drugim zaposlenim, ki bi se želeli mimogrede informirati o stanju nalog ter trenutnem delu razvojne ekipe. V elektronski obliki stanje in dogajanje na nalogah spremljamo zaradi dolgoročnih ciljev, da se bo skozi čas ter več intervalov ocenilo, koliko nalog je možno opraviti v posameznem intervalu. Ključno je, da je ocenjevanje trajanja nalog čimbolj realno ter skupna ocena, kdaj bo projekt ali skupina posameznih nalog opravljena, čim natančnejša.

Poleg agilnega modela razvoja programske opreme se v obravnavanem primeru izvaja tudi tradicionalni management projektov. Uporaba obeh omogoča celovit pregled napredka ter pravočasno ukrepanje z aplikacijo tehnik skrajševanja časovnice, vzporednim izvajanjem nalog, ko razpoložljivost virov to dopušča ter s prerazporeditvijo virov.

Ocenjujem, da je vpeljava agilnega modela razvoja, poleg splošnega dviga zavesti o pomenu kakovosti, največja pridobitev razvojnega oddelka. Vpeljava modela omogoča bolj sistematično in kakovostno delo ter boljši pregled nad izvajanjem nalog. Predvsem pa trajen zapis o zmogljivostih razvojne ekipe oz. kako učinkovito in hitro je izvajanje razvojnih nalog. Poleg tega povzroči, da se nobena nižje prioritetna aktivnost ne izvaja brez vedenja, koliko časa in virov zahteva.

7.4.3 Uvedba aktivnosti za izboljšanje kakovosti programske opreme

Poleg managerskih so bile v preteklih mesecih v razvojni proces uvedene nekatere inženirske izboljšave, za katere menim, da pripomorejo k splošnemu izboljšanju dela tako z vidika kakovosti kot tudi učinkovitosti dela.

Povečanje kakovostne komunikacije pomeni posledično bolje izbrane funkcije programskih modulov. V uporabi je metoda možganskega viharjenja (angl. *Brain Storming*), ki pripomore, da se vsi udeleženci širše zavedajo ciljev razvoja programskih modulov in tudi povezav do povezanih programskih modulov. Ocenjujem, da diskusija v kateri sodelujejo različni profili ter odgovorni za posamezne programske sklope, privede do kakovostnejših programskih modulov. Rezultat je ustreznejša programska rešitev, z zmanjšano stopnjo tveganja, da bo potrebno ponovno delo ali modifikacija med izvedbo, kar zagotavlja učinkovitejše delo razvojne ekipe in boljšo rabo časa.

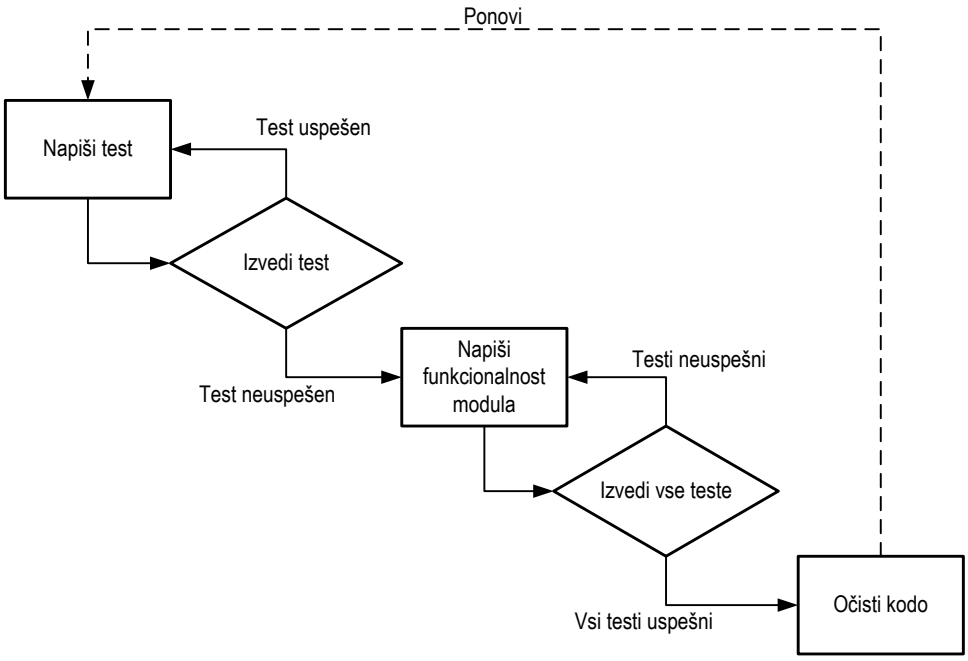
Redno se izvaja pregledovanje kode s strani sodelavcev (angl. *Peer Review*), predvsem modulov, ki so na novo razviti in bistveno vplivajo za povezane module. Pregledovanje se izvaja v paru izvajalec-pregledovalec, kot nadomestek programiranja v paru (angl. *Pair Programming*), ki se le redko izvaja. Učinek pregleda je boljše zavedanje o razvitih rešitvah in hkrati se oba programerja seznanita s pogledi drugega in vzroki za posamezne rešitve. S tehniko odkrijemo in odpravimo veliko število hroščev, ki jih je težje najti v procesu testiranja.

Problem količine testiranja, za katero ocenjujem, da ga je bilo v preteklosti premalo, rešujemo že na nivoju testov enot. V razvojnem oddelku smo uvedli pristop TDD (angl. *Test Driven Development*) in hkrati nadgradili dosedanji način dela z izboljšanjem kakovosti dokumentacije o delovanju ter rešitvah programskih modulov. Uveljavljena sta 2 programerska pristopa, izbrana glede na kompleksnost posameznih modulov:

- Pristop TDD. Pomeni drugačen pristop k izdelavi programskih modulov. Razvojni inženir prične razvojni cikel programskega modula s pisanjem kode testov enot. Testi morajo ob prvi izvedbi pasti oz. biti neuspešno izvedeni, ker funkcionalne programske rešitve še ni. Inženir delo nadaljuje s pisanjem kode, ki predstavlja funkcionalnost programske rešitve. Razvoj se nadaljuje z iterativnim izvajanjem testov in popravljanjem funkcionalnega dela kode. Ko so vsi testi uspešno izvedeni oz. je njihov rezultat pozitiven, je programski modul funkcionalno ustrezen. Sledi še čiščenje kode nepotrebnih rutin ter predaja v integracijo.
- Drugi pristop je MDSD (angl. *Model Driven Software Development*), ki temelji na uporabi modelov za zasnovo in razumevanje delovanja programskih modulov. Pristop je postal popularnejši po uveljavitvi enotnega modelirnega jezika UML (angl. *Unified Modeling Language*), ki je postal standard za modeliranje programske kode. MDSD pristop je pomemben, ko je potrebno pred začetkom kodiranja programske rešitve

zagotoviti razumevanje rešitve tudi netehničnim kadrom. Delujoče programske rešitve še ni, vendar z modeli nazorno predstavimo rešitve.

Slika 13: TDD pristop k razvoju programskih modulov



Vir: Test driven development, b.l.

Presoja kakovosti programske opreme

Glede razumevanja pomena karakteristik kakovosti programske opreme se v podjetju uporablja standard ISO 9126, ki služi kot osnovno orodje za preučevanje, katere kakovostne karakteristike je potrebno izboljševati.

Po analizi standarda sem opravil presojo kakovosti programske opreme. Razgradnja zahtev glede kakovosti programske opreme loči zahteve na (Fajfar & Benčina, 2006, str. 20):

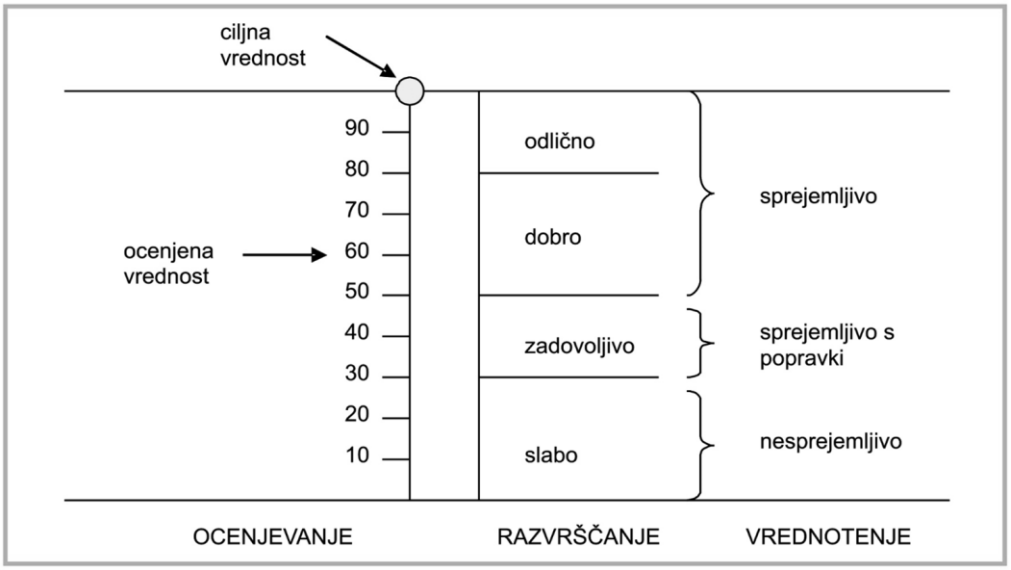
- zahteve glede opisa programske opreme,
- zahteve glede dokumentacije programske opreme ter
- zahteve programov in podatkov.

Prvi dve kategoriji obravnava standard ISO 9127. Opravil sem presojo glede zadnje kategorije, samega programa oz. kode. Postopek presoje kakovosti se vrši v korakih ocenjevanje, razvrščanje in vrednotenje z vidika karakteristik in podkarakteristik standarda ISO 9126, in obsega 5 korakov:

1. Izvedba dekompozicije izbranih karakteristik do nivoja podrobnosti, da bo mogoča meritev v metrični ali nominalni skali. Zaradi preglednosti ohranimo vseh 6 področij kakovosti.

2. Opredelimo ciljne vrednosti karakteristik. Ciljna vrednost temelji na subjektivni oceni ocenjevalca.
3. Izvedemo merjenje in ocenjevanje.
4. Rezultate merjenja uvrstimo v skalo po kriteriju, kako blizu je ocenjena vrednost ciljni vrednosti.
5. Ovrednotimo rezultat.

Slika 14: Lestvica ocenjevanja kakovosti



Vir: P. Fajfar & J. Benčina, Napotki za presojo kakovosti programskih proizvodov, 2006, str. 21.

Postopek presoje, ki sem ga opravil, da sledeče rezultate:

- Funkcionalnost ocenjujem z oceno dobro. Karakteristiko funkcionalnosti ocenjujem z visoko oceno, saj ima naprava številne uporabnikom prilagojene lastnosti. Kljub temu je stalna skrb, da se obstoječe stanje ohrani nujna.
- Zanesljivost ocenjujem s povprečno oceno zadovoljivo. Naprava deluje zanesljivo v veliki večini situacij v katerih je uporabljena. Kljub temu se v manj ugodnih pogojih zgodi, da ne more zagotoviti rezultatov. Karakteristiko bomo izboljšali z uvedbo dodatnih aktivnosti povezanih s kakovostjo programske opreme, predvsem z dodatnim testiranjem in spremljanjem programskih hroščev.
- Uporabnost ocenjujem z oceno odlično. Programska oprema je enostavna in njene uporabe se je možno hitro priučiti.
- Učinkovitost ocenjujem z oceno dobro. Programska oprema je odzivna v realnem času in za delovanje ne potrebuje velike količine virov glede strojne opreme.
- Vzdrževalnost ocenjujem z oceno zadovoljivo. V prihodnosti bo potrebno lastnost vzdrževalnosti izboljšati, da bo odkrite napake možno hitreje odpraviti ter se po drugi strani hitreje odzvati na želene spremembe delovanja s strani uporabnikov. Karakteristiko vzdrževalnosti bomo povečali z ureditvijo programske kode v module

oz. knjižnice, v bolj kompaktno programsko kodo, ki bo manjša, uvedbo konfiguracij ter drugimi tehničnimi izboljšavami.

- Prenosljivost ocenjujem z oceno slabo. Nameščanje programske opreme je sicer enostavno, možno tudi po tem, ko je naprava že montirana v okolju uporabe. Programska koda ni v skladu z izbranimi standardi. Potreben bi bil znaten napor, da bi kodo uskladili s standardi kodiranja.

7.5 Model TMMI

Kot odgovor testne skupnosti na modela zrelostnih stopenj CMMI, je bil razvit zrelostni model, ki je osredotočen na področje testiranja. Lestvica modela opredeli 5 zrelostnih stopenj (Ammann & Offutt, 2008, str. 8; TMMI Foundation, 2012, str. 9). Opisi stopenj so:

- Stopnja 1: Testiranje je kaotično in je večkrat obravnavano kot del razhroščevanja. Testiranje je odvisno od posameznikov, saj organizacija ne omogoča stabilnega okolja za testiranje. Organizacija ne zagotavlja vseh virov ter ne skrbi za usposobljenost izvajalcev testiranja. Testiranje je izvedeno po lat. *ad-hoc* pristopu, takoj ko je zaključeno kodiranje. Njegov namen je pokazati, da programska rešitev deluje oz. je brez najtežjih hroščev. V tej stopnji ni zagotovljenih virov za testiranje tj. opreme ter predvsem ustreznega kadra.
- Stopnja 2: Situacija je manj kaotična kot v prejšnji stopnji. Testiranje je jasno ločeno od procesa razhroščevanja programske kode, vendar je s strani deležnikov projekta še vedno obravnavano kot faza, ki sledi kodiranju. V tej fazi se dvigne zavest pomena testiranja in med razvojem se pripravi testne načrte ter izbere pristop k testiranju. Ločuje se tudi različne nivoje testiranja, enotne, integracijske in sistemske teste.
- Stopnja 3: Testiranje se uveljavi na nivoju organizacije. Testiranje je integrirano v razvojni proces in je povezano z doseganjem mejnikov projekta. Testiranje je opredeljeno v zgodnji fazi managementa projekta in je dokumentirano v projektni dokumentaciji. Testerji so vključeni v pregledovanje zahtev ter specifikacij bodočega produkta. Organizacija skrbi za ustrezno usposobljenost testne ekipe in testiranje je obravnavano kot poklic. Organizacija se tudi zaveda pomena kakovosti ter pomena testiranja v procesu zagotavljanja ter kontrole kakovosti.
- Stopnja 4: S sumiranjem vseh nižjih stopenj je testiranje na stopnji 4 merljiv proces. Uvedene so metrike testiranja in testiranje je obravnavano kot ocenjevalec procesa razvoja. Testiranje je uvedeno v najzgodnejše razvojne faze in njegovi rezultati so faktor pri odločanju o poteku projekta, ki temelji na merljivih dejstvih. Uvedene so naprednejše tehnike, inšpekcije, pregledi kode ipd. Posebna skrb je posvečena izboljšanju učinkovitosti testiranja.
- Stopnja 5: Najvišja stopnja. Testiranje je v popolnost definiran proces. Organizacija deluje na stalnem izboljševanju stopnje 4 preko osvajanja novih tehnologije ter tehnik testiranja. Organizacija ima poleg posebne testne ekipe, že ustanovljeno tudi ekipo za izboljšave testnih procesov, ki ji, glede na prejšnje stopnje, raste odgovornost. Posebna

pozornost se posveča usposobljenosti obeh ekip. Identificira ter analizira se najpogostejše vzroke za napake ter se ustrezno ukrepa, da se ti ne bi ponavljali v prihodnosti.

7.5.1 Analiza procesa testiranja

Analiza obravnavanega podjetja, katere cilj je oceniti, v katero stopnjo podjetje spada, je ob začetku uvedbe sprememb v razvojni proces, podjetje umestila v stopnjo 1. Poglavitni razlogi za umestitev v stopnjo 1 so:

- Proces testiranja ni jasno opredeljen. Nivoji testiranja so slabo določeni in testiranje višjih nivojev se večkrat ne izvaja. Testiranje je v obliki razhroščevanja vpeto v kodiranje.
- Predvsem višje nivojski testi oz. integracijsko in sistemsko testiranje se izvajata v premajhnem obsegu, medtem ko je testov na nivoju enot več. Kasneje prevzemno testiranja razgali hrošče, ki se ne bi smeli pojaviti pri stranki.
- Testna dokumentacija je pomanjkljiva. Testni načrt, opis testnih primerov in testnih procedur ne obstaja ali pa so opisi nenatančno dokumentirani ali ne izhajajo iz specifikacij programskih modulov.
- Podjetje nima stalne testne ekipe in testerjev specialistov, ki bi posedovali ustrezna znanja testiranja in bili sposobni dvigniti kakovost testnih aktivnosti.

Skozi analizo izvajanja razvojnih projektov smatram, da se je pomenu testiranja pripisovalo premajhno vlogo in posledično izvajalo premalo testiranja. Po preučevanju modela TMMI ocenjujem, da bi za pridobitev stopnje 2, bilo potrebno:

- Spremeniti odnos do testiranja in posledično definirati proces testiranja kot posebno fazo v procesu razvoja programske opreme.
- V fazi planiranja za aktivnosti testiranja rezervirati nekajkrat več časa. Avtorji s področja razvoja programske opreme ter testiranja različno ocenjujejo potreben delež časa razvoja, ki ga je potrebno nameniti testiranju. Vsi se strinjajo, da mora biti tega časa vsaj 30%.
- Zagotoviti vire potrebne testnim aktivnostim. Testna oprema je na voljo v ustreznem obsegu. Razpoložljivost človeških virov zaradi težavnosti ocenjevanja trajanja testnih aktivnosti in zaradi zasedenosti inženirjev z drugimi aktivnostmi, predstavlja večji problem.
- Zagotoviti ustrezno testno dokumentacijo, ki omogoča sistematičen pristop k testiranju. Narediti je potrebno naslednjo dokumentacijo testni načrt, opis testnih primerov in opis testnih procedur. Kot osnova za izdelavo testne dokumentacije je možna uporaba standarda IEEE 829.
- Vpeljati sistem za sledenje napakam, ki bo zagotovil celovit pregled nad stanjem programske kode ter omogočal pravočasno odpravo napak ter ustrezno potrjevanje popravkov.

V času po začetku vpeljave sprememb so bile na področju testiranja uvedene spremembe, katerih rezultat je, da splošen opis testnih aktivnosti ustreza opisu stopnje 2. Vpeljane spremembe so opisane v nadaljevanju.

7.5.2 Uvedba sprememb v proces testiranja

Povečanje obsega testnih aktivnosti

Po spremembah se v podjetju dokaj vestno izvaja vse nivoje testiranja, tudi regresijsko testiranje, ko se za to identificira potreba. Testiranje višjih nivojev, npr. sistemsko ter prevzemno testiranje, sledi fazi kodiranja.

Povečan obseg testnih aktivnosti smatram kot ustrezen odgovor na probleme povezane z nekakovostjo, ki že daje rezultate. Omogoča nam identifikacijo hroščev v zgodnji fazi razvoja programske opreme ter preprečuje stroške, ki bi nastali z zamenjavami in vpoklicem že montiranih naprav ali z nadgradnjami programske opreme na terenu.

Možnost izvajanja vseh testnih nivojev pripisujem tudi boljšemu managementu časa in ustreznem ukrepanju, ko je projekt v zamudi. Na ta način za testne aktivnosti ostane na razpolago dovolj časa in izvajanje testnih aktivnosti ne vpliva na datum zaključka projektov.

Izdelava testne dokumentacije

Razvojna ekipa je v procese vpeljala najboljše prakse, ki jih opredeljujejo standardi s področja razvoja programske opreme, s ciljem, da na daljši rok dvigne nivo kakovosti in učinkovitosti svojega dela. Uvedeni so standardi glede dokumentiranja dizajna rešitve programske opreme IEEE 1016 in glede testiranja ter izdelave testne dokumentacije po IEEE 829.

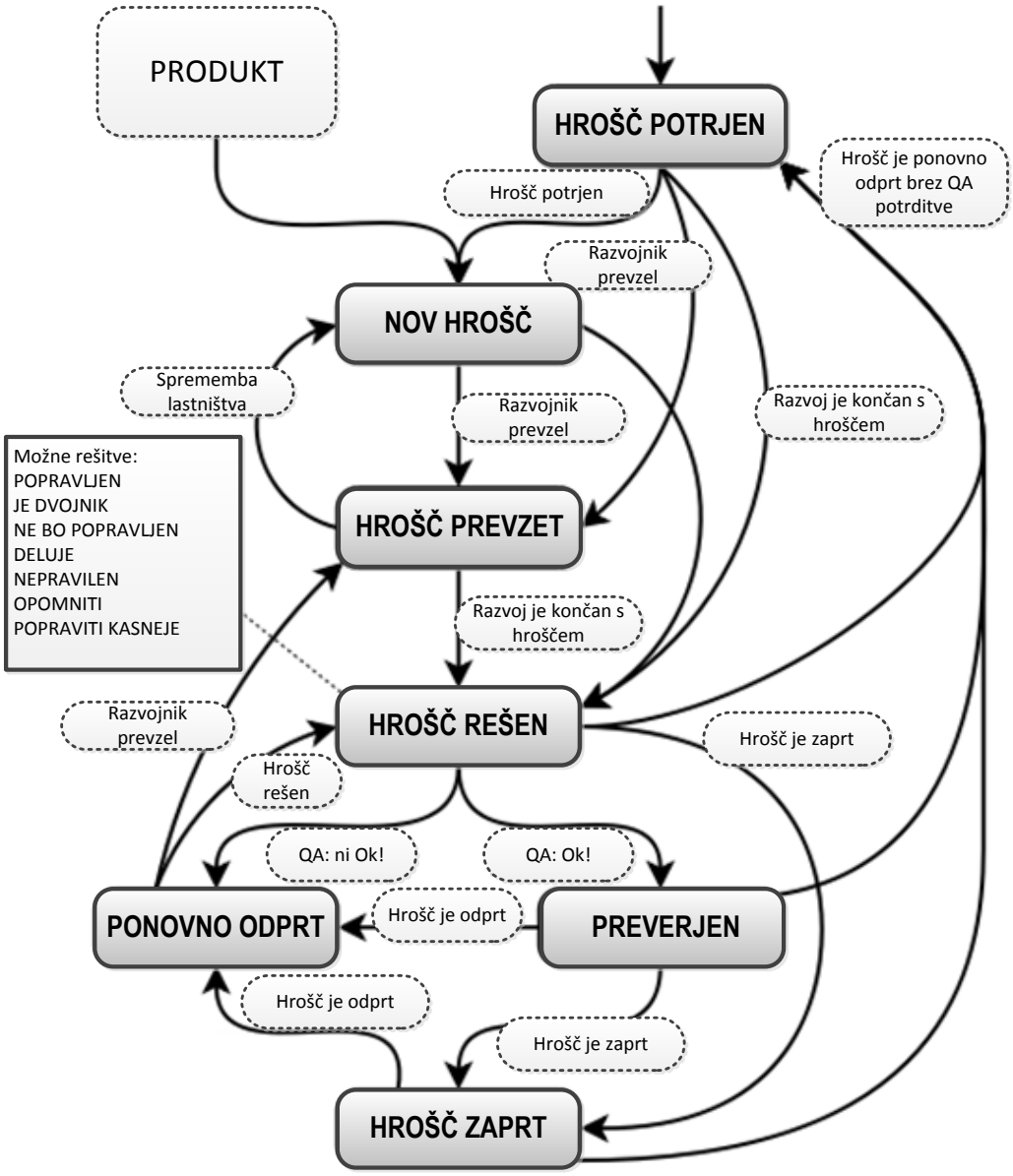
Za potrebe testiranja so bili izdelani dokumenti, testni načrt, opis testnih primerov in podroben opis testnih procedur. Testni postopki temeljijo na podrobno opisanih funkcionalnostih programskih modulov. Teste je mogoče ponoviti po testnih procedurah in se pri opisu programskih hroščev sklicevati na opisan testni primer oz. proceduro.

Ekipa pričakuje manj napak v programski kodi, manj ponovnega dela ter učinkovitejše vzdrževanje produktov kot posledico boljše in natančnejše dokumentacije.

Sistem za sledenje hroščem

V procesu testiranja ocenjujem, da je nujna uporaba orodij za sledenje hroščem (angl. *Bug Tracking System*). Prepričanje sledi iz preteklih izkušenj ter opisov najboljših praks, ki so na voljo na spletu ter v drugih virih.

Slika 15: Življenjski cikel programskega hrošča



Vir: The Bugzilla Team, The Bugzilla Guide, 2006.

Sistem omogoča vpogled v delo in produktivnost ekipe programerjev ter testerjev preko vpogleda v dinamiko odkrivanja ter popravljanja hroščev. Metrike, kot so število hroščev in njihova teža, omogočajo odgovornim, da ocenijo kakovost programske opreme in sprejmejo odločitev o najugodnejšem trenutku za izdajo produkta na trg oz. naročniku.

Med testiranjem sem kot član razvojne ekipe in občasno testne ekipe, močno vzpodbujal uporabo programske opreme za spremljanje življenjskega cikla hroščev, ki je nepogrešljiv del vsakega procesa razvoja programske opreme. Menim, da uporaba sistema velja za znak zrelega delovanja razvojne in testne ekipe ter omogoča programerjem ter testerjem učinkovito spremljanje rezultatov aktivnosti testiranja.

Sistem za sledenje hroščem je v našem primeru integriran v programsko opremo za management projektov razvoja programske opreme. Pomembna lastnost sistema je, da podpira princip življenjskega cikla hrošča.

Tipičen življenjski cikel je identifikacija in prijava hrošča, popravljanje hrošča ter verifikacija popravka. Pri tem sodelujeta v procesu vsaj 2 strani. Testna ekipa je stran, ki hrošč prijavi ter po popravku verificira oz. potrdi ustreznost rešitve ter stran, ki hrošč odpravi, npr. programerji, razvojniki.

Uporaba sistema je zagotovila, da so se odkriti programski hrošči sproti celovito beležili in so bili urejeni in popravljeni po prioriteti, glede na resnost posledic, ki bi jih imeli na delovanje naprave.

7.6 Predlogi nadaljnjih izboljšav

Kljub uvedenim spremembam, ki že imajo pozitiven učinek, bo potrebno nadalje izpopolnjevati procese na zelo širokem spektru področij. Za osnovo je mogoče ponovno uporabiti CMMI ter TMMI modela in se v prihodnosti posvetiti področjem, ki omogočajo nadaljevanje vzpona po lestvici stopenj modela. V bližnji prihodnosti zato predlagam vpeljavo naslednjih ukrepov.

Vpeljava funkcije projektnega managementa

V primeru, da bo podjetje raslo ali povečalo število projektov, bo potrebno vpeljati funkcijo projektnega managementa. V tem trenutku vlogo projektnega managerja še vedno ne opravlja specialist projektni manager, temveč enako kot prej vodilni inženir. Sicer bolj natančno, saj se je zavedanje o potrebnosti tega dela dvignilo.

Na področju projektnega managementa bo potrebno izboljšati management časa oz izboljšati časovno planiranje preko izboljšanja ocenjevanja trajanja aktivnosti. Menim, da je ocenjevanje na podlagi ekspertnih metod in izkustev ustrezno, vendar ga je potrebno nadgraditi z vpeljavo merljivih dejstev, npr. opravljenih ur, številom osebja ipd.

Vpeljava testne ekipe

Na organizacijskem področju bo potrebno organizirati ter izobraziti stalno testno ekipo. Njena naloga bo, poleg testiranja, tudi skrb za izdelavo kakovostne testne dokumentacije na osnovi specifikacij naprave, v obliki testnih načrtov, testnih primerov ter izdelava testnih poročil ob koncu testiranja.

Potrebno bo vztrajanje na uporabi testnih orodij za sledenje hroščem odkritih med testiranjem ter tudi med uporabo pri stranki.

Avtomatizacija testiranja

Za časovno učinkovitost opravljanja testnih aktivnosti je ključna avtomatizacija testov, ki omogoča hitro izvajanje testiranja na vseh nivojih ter olajša izvedbo regresijskih testov. Podjetje že razvija lastno testno infrastrukturo za prototipna testiranja ter testiranja proizvodne serije naprav.

Ker je testiranje proces, ki zahteva veliko časa ter človeških virov, z avtomatizacijo prihranimo oba vira. Praviloma si želimo avtomatizirati vse teste, ki jih je mogoče (Ammann & Offutt, 1999, str. 10).

Analiza končanih projektov

V zaključni fazi projektov je v nekaterih podjetjih vsem deležnikom projekta naročeno, naj pripravijo kratek povzetek svojega videnja projekta.

Poročila so uporabljena v naslednjih projektih kot vir znanja, ki omogoča, da se na prihodnjih projektih ne ponavljajo stare napake. Poročila so tudi pomoč pri boljšem planiranju novih projektov in z njimi sežemo na področje managementa tveganj in na področje zagotavljanja kakovosti.

SKLEP

Projekt je skupina povezanih aktivnosti, ki vključuje uporabo različnih znanj in veščin v vseh življenjskih fazah projekta. Za izvedbo projektov razvoja programske opreme je potrebno znanje projektnega managementa in na pravi način uporabiti inženirska znanja izdelave programske opreme.

V ospredju sta management časa ter management kakovosti, ki sta področji, ki jih projektni managerji, predvsem v manjših podjetjih, intenzivno izvajajo. Management časa je pomemben zaradi pravočasne izvedbe projekta ter pravočasnega nastopa produkta na trgu. Management kakovosti pa zagotovi, da je produkt skladen z zahtevami projekta. Komponenti sta povezani in v določeni meri nasprotujoči. Kakovost zahteva svoj čas, a hkrati je pomembno tudi izvajanje projekta v skladu s časovnim načrtom projekta.

Management časa ter kakovosti sem obravnaval na primeru podjetja, ki je razvijalec programske opreme za zelo specifično uporabo in je hkrati razvijalec in proizvajalec tudi strojne opreme naprav. Opazoval sem izvajanje aktivnosti razvoja programske opreme ter analiziral glavne pomanjkljivosti procesa. Pomanjkljivosti identificiram v neizvajanju ali površnem izvajanju nekaterih faz projekta. Predvsem očitno je bilo pomanjkanje planiranja. Posledično pa trpijo tudi druge faze in aktivnosti.

Management časa je pomemben iz očitnega razloga, ker uspešen, pomeni krajši čas trajanja projekta in poleg tega, da je produkt na trgu prej, tudi manj stroškov. Pomemben je že na začetku projekta, med planiranjem, ko je ključno, da je ocenjeno trajanje aktivnosti realno

in bo privedlo do realne in uresničljive časovnice. Le s takšno bo mogoče v fazi izvedbe učinkovito kontrolirati čas ter opravljati korektivne ukrepe.

Vzroki za neizvajanje planiranja so v splošnem nepoznavanje ter neusposobljenost kadrov za tovrstno delo. Manjša tehnološka podjetja si zaradi stroškov težje privoščijo specialiste z znanji projektnega managementa in so največkrat sestavljena iz specialistov inženirjev. Ker popolnoma brez projektnega načina dela vendarle ne gre, so posamezni tehnični kadri prisiljeni opravljati delo projektnega managerja. Tem kadrom je to sekundarno delo. Njihovo primarno delo ostaja opravljanje nalog s tehničnega področja.

V obravnavanem primeru je bil poseben izziv prepričati deležnike projekta o potrebnih daljših časih za izvedbo projektov oz. nalog, ki so bili realnejši in obenem omogočili višjo končno kakovost produkta. Prepričanje, da so aktivnosti povezane s kakovostjo potrebne, bo še dodatno zraslo na daljši rok, ko se bodo pokazali pozitivni učinki tudi povečanega obsega testiranja, preko manj odpovedi programske opreme, manj hroščev ter posledično manj vzdrževanja po koncu projekta.

Enako pomemben je tudi management kakovosti. V podjetju se bolje kot prej zavedamo, kaj pomeni kakovost programske opreme oz. smo se sposobni osredotočiti na posamezne karakteristike kakovosti, predvsem na tiste, ki so po analizi pridobile višji pomen.

Proces zagotavljanja kakovosti ne smemo enačiti ali zamenjevati s procesom kontrole kakovosti. Proces zagotavljanja kakovosti pomeni implementirati potrebne izboljšave za preprečevanje napak. V primeru razvoja programske opreme gre v prvi vrsti za tehnične ter inženirske izboljšave, napredek ter spremembe nekaterih organizacijskih ter kulturnih vidikov podjetja.

Kontrola kakovosti je lažje razumljiv pojem, največkrat ga enačimo s testiranjem. Ključno je, da proces testiranja opravljamo sistematično in v skladu z najboljšimi praksami. *Ad-hoc* pristop zaradi kompleksnosti programske opreme ni dovolj in je neučinkovit. Le v primeru, da se testiranje izvaja v vseh fazah razvoja in se ne preskakuje nivojev testiranja, lahko pričakujemo v končnem produktu malo napak, saj se bodo le-te odkrile in odpravile dovolj zgodaj in ne bodo odkrite pri masovni uporabi programske opreme, torej pri stranki. V primeru, da se napake odkrijejo pri stranki, podjetje utrpi tako trenutne kot prihodnje finančne posledice, kot tudi nemerljivo izgubo ugleda.

Razvojni ekipi so v pomoč tudi predstavljeni standardi kakovosti, ki se postopoma vpeljujejo v razvojne aktivnosti. Podjetje pričakuje, da bo z dvigom kakovosti na dolgi rok znižalo stroške vzdrževanja produktov, lažje pridobivalo naročila ter povečalo ugled pri strankah.

V praktičnem delu naloge sem poskusil obe področji povezati ter analizo delovanja podjetja in kasnejše uvedene spremembe povezati z zrelostnima modeloma CMMI ter

TMMI. Cilj analize ter vpeljanih sprememb je bil v obeh modelih pridobiti eno stopnjo, kar je bilo delno doseženo.

Menim, da je zadan cilj naloge, izboljšati način dela ter dvigniti kakovost izdelkov, tudi preko boljšega managementa časa, pretežno dosežen. Mnogo izboljšav je implementiranih v proces, predvsem pa menim, da zgledi vlečejo in bo v prihodnosti v podjetju možno lažje in hitreje vpeljevati nove izboljšave.

LITERATURA IN VIRI

1. Ammann, P., & Offutt, J. (2008). *Introduction to Software Testing*. New York: Cambridge University Press.
2. Beck, K. (2003). *Test Driven Development by Example*. Boston: Pearson Education
3. The Bugzilla Team. (2006). *The Bugzilla Guide - 2.18.6 Release*. Najdeno 21. septembra 2013 na spletnem naslovu http://www.bugzilla.org/docs/2.18/html/lifecycle.html
4. Fajfar, P., & Benčina, J. (2006). Napotki za presojo kakovosti programskih proizvodov. *Kakovost SZK,* (2), 17 – 22.
5. Fuller, M., Wallach, J. S., & George, J. F. (2008). *Information Systems Project Management, A Process and Team Approach*. b.k.: Prentice Hall.
6. Galin, D. (2004). *Software Quality Assurance, From Theory to Implementation*. Edinburgh Gate: Pearson Education Limited.
7. Haughey, D. (2013). Understanding the Project Management Triple Constraint. Najdeno 4. maja 2013 na spletnem naslovu http://www.projectsmart.co.uk/understanding-the-project-management-triple-constraint.html
8. *The Importance of Knowing Your Critical Path*. Najdeno 10. februarja 2013 na spletnem naslovu http://www.articlesbase.com/project-management-articles/the-importance-of-knowing-your-critical-path-1320876.html
9. Institute of Electrical and Electronics Engineers. (2008). *IEEE 829-2008 Standard for Software and System Test Documentation*. New York: The Institute of Electrical and Electronics Engineers.
10. Institute of Electrical and Electronics Engineers. (2009). *IEEE Std 1016-2009 Standard for Information Technology – Systems design – Software design description*. New York: Institute of Electrical and Electronics Engineers.
11. International Standards Organization/International Electrotechnical Commission. (2000). *ISO 9126 Information Technology - Software Product Quality*. b.k.: International Standards Organization.
12. International Standards Organization/International Electrotechnical Commission. (2008). *ISO 12207 Systems and software engineering – Software life cycle processes*. New York: The Institute of Electrical and Electronics Engineers.
13. International Standards Organization/International Electrotechnical Commission. (2013). Principles for Developing ISO and IEC Standards Related to or Supporting Public Policy Initiatives. Najdeno 10. marca 2013 na spletnem naslovu http://www.iso.org/iso/principles_for_developing_iso_and_iec_standards_related_to_or_supporting_public_policy_initiatives.pdf
14. Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The Unified Software Development Process*. Reading, Massachusetts: Addison-Wesley.
15. Kampuš, A. (2002). *Projektni management pri razvoju programskih rešitev* (magistrsko delo). Ljubljana: Ekonomska fakulteta.

16. Kelemen, M. L. (2003). *Managing Quality*. London: SAGE Publications Ltd.
17. Kerzner, H. (2003). *Project Management: A System Approach to Planning, Scheduling and Controlling* (8th ed.). b.k.: Wiley Publishing.
18. Lewis, J. P. (1995). *Fundamentals of Project Management*. b.k.: AMACOM Books.
19. Lipke, W. (2012). *Earned Schedule*. b.k.: lulu.com.
20. *Literate programming and the rational unified process*. Najdeno 3. julija 2013 na spletnem naslovu http://cs460-thaumkid.blogspot.com/2011/03/literate-programming-and-rational.html
21. Lock, D. (2007). *Project Management* (9th ed.). Hampshire, England: Gower Publishing Ltd.
22. Naik, S., & Tripathy, P. (2008). *Software Testing and Quality Assurance: Theory and Practice*. New Jersey: Wiley.
23. Patton, R. (2006). *Software Testing* (2nd ed.). b.k.: SAMS Publishing.
24. Peters, L. J. (2008). *Getting Results From Software Development Teams*. Redmond Washington: Microsoft Press.
25. Povodnik, J. (2012). *Management kakovosti pri projektih razvoja programske opreme* (magistrsko delo). Ljubljana: Ekonomska fakulteta.
26. Project Management Institute. (2008). *Vodnik po znanju projektnega vodenja* (tretja izdaja). Kranj: Moderna Organizacija.
27. Project Management Institute Standards Committee (2004). *A Guide to the Project Management Body of Knowledge* (3rd ed.). b.k.: Project Management Institute.
28. Schwaber, K. (2004). *Agile Project Management with Scrum* (1st ed.). Redmond, Washington: Microsoft Press.
29. Schwalbe, K. (2010). *IT Project Management*. Boston: Course Technology.
30. Silva, N. (2012). 5 Reasons to Use Gantt Charts. Najdeno 10. septembra 2013 na spletnem naslovu http://creately.com/blog/diagrams/5-reasons-to-use-gantt-charts/
31. Software Engineering Institute. (2010). *CMMI for Development*. b.k. Carnegie Mellon University.
32. *Software Testing*. Najdeno 10. maja 2013 na spletnem naslovu http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/
33. Solina, F. (1997). *Projektno vodenje razvoja programske opreme*. Ljubljana: Založba FE in FRI.
34. Šajtegel, V. (2003). *Analiza uspešnosti organiziranja za projektni management v podjetju Hermes Softlab* (magistrsko delo). Ljubljana: Ekonomska fakulteta.
35. *Ten Key factors for ensuring successful Release Management*. Najdeno 3. marca 2013 na spletnem naslovu http://www.bluefinsolutions.com/insights/blog/10_kay_factors_for_ensuring_successful_release_management/
36. *TenStep - The Value of Project Management*. Najdeno 10. aprila 2012 na spletnem naslovu http://www.tenstep.com/open/A1ValueofPM.html
37. Test driven development. (b.l.). v Wikipedia. Najdeno 10. septembra 2013 na spletni strani http://en.wikipedia.org/wiki/Test-driven_development

38. TMMi Foundation. (2012). *Test Maturity Model Integration, Release 1.0*. b.k.: TMMi Foundation.
39. Vrhovšek, R. (2005). *Analiza vodenja projektov razvojnem procesu Iskre Transmission* (magistrsko delo). Ljubljana: Ekonomska fakulteta.
40. Young, T. L. (2007). *The Handbook Of Project Management*. London: Kogan Page.
41. Živković, S. (2011). *Ocenjevanje obsega dela pri projektih razvoja programske opreme* (magistrsko delo). Ljubljana: Ekonomska fakulteta.