UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

MAGISTRSKO DELO

MANAGEMENT KAKOVOSTI PRI PROJEKTIH RAZVOJA PROGRAMSKE OPREME

Ljubljana, oktober 2012 JANJA POVODNIK

IZJAVA O AVTORSTVU

Spodaj podpisani(-a) Janja Povodnik, študent(-ka) Ekonomske fakultete Univerze v Ljubljani, izjavljam, da sem avtor(-ica) zaključne strokovne naloge/diplomskega dela/specialističnega dela/magistrskega dela/doktorske disertacije z naslovom Management kakovosti pri projektih razvoja programske opreme, pripravljene(-ga) v sodelovanju s svetovalcem/svetovalko prof. dr. Talibom Damijem in sosvetovalcem/sosvetovalko _____.

Izrecno izjavljam, da v skladu z določili Zakona o avtorskih in sorodnih pravicah (Ur. l. RS, št. 21/1995 s spremembami) dovolim objavo zaključne strokovne naloge/diplomskega dela/specialističnega dela/magistrskega dela/doktorske disertacije 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(-a), da so dela in mnenja drugih avtorjev oziroma avtoric, ki jih uporabljam v zaključni strokovni nalogi/diplomskem delu/specialističnem delu/magistrskem delu/doktorski disertaciji, citirana oziroma navedena v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani, in
    - pridobil(-a) 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(-a);
- se zavedam, da je plagiatorstvo – predstavljanje tujih del (v pisni ali grafični obliki) kot mojih lastnih – kaznivo po Zakonu o avtorskih in sorodnih pravicah (Ur. l. RS, št. 21/1995 s spremembami);
- se zavedam posledic, ki bi jih na osnovi predložene zaključne strokovne naloge/diplomskega dela/specialističnega dela/magistrskega dela/doktorske disertacije dokazano plagiatorstvo lahko predstavljalo za moj status na Ekonomski fakulteti Univerze v Ljubljani v skladu z relevantnim pravilnikom.

V Ljubljani, dne _____ Podpis avtorja(-ice):_____

KAZALO

KAZALO SLIK

UVOD

Danes si življenja brez računalnikov ne predstavljamo več, kar posledično pomeni vse večje potrebe po različni programski opremi (angl. *Software*) in vse bolj zmogljivi računalniški oz. strojni opremi (angl. *Hardware*). S tem postajajo projekti informacijske tehnologije (v nadaljevanju IT) in njihovo managiranje v družbi ključnega pomena. Podjetja se ukvarjajo z razvojem različnih področij v IT, ki so vezana predvsem na želje in potrebe odjemalcev, kupcev in naročnikov. Bolj kot nek izdelek ustreza željam potrošnikov, večji uspeh bo požel. Pri programski opremi, na katero se bomo v magistrskem delu tudi osredotočali, je čedalje pomembnejša kakovost razvoja, izvajanja, delovanja itd. programske opreme. Skozi delo bomo spoznali, kako pomembno je v svoj načrt izvedbe vpeljati kakovost in jo preverjati skozi cel projekt.

Na področju informatike se navadno izvajata razvoj in uvajanje novosti skozi delo, ki je organizirano v projekte. Poslovni projekti so tudi projekti IT (projekti informacijske tehnologije). Od večine ostalih poslovnih projektov se razlikujejo po tem, da v svoje izvajanje vključujejo informacijsko tehnologijo. Projekti IT lahko pomembno prispevajo k uresničevanju strategije podjetja, torej jih je potrebno povezati s strateškimi cilji in njihovimi merljivimi kazalci. Obravnavati jih moramo kot vsako drugo poslovno investicijo, torej z izračunom o donosu projekta. IT tako odgovarja za razvoj ustrezne programske opreme, medtem ko je ostalo odvisno od podjetja, ki je programsko opremo naročilo in njihovega vpeljevanja v obstoječo strukturo, zmogljivosti strojne opreme itd. K uvedbi nove programske opreme veliko prispeva tudi pripravljenost zaposlenih za pridobitev novih znanj, ki jih potrebujejo za uporabo le-te.

Čibej (2010, str. 11–12) meni, da je vsakdanjost nekoliko drugačna, saj na splošno velja, da veliko informacijskih projektov ne uspe. Največkrat za to situacijo krivijo premajhno zavzetost vrhnjega managementa in pomanjkljivo analizo potreb. Vse preveč informacijskih projektov se začne na osnovi nerealističnih pričakovanj, ki jih je kasneje iz različnih razlogov nemogoče spremeniti. Projektni managerji morajo tako sprejeti terminske načrte, do katerih so skeptični, in računajo, da bo med izvajanjem projekta prišlo do temeljite spremembe načrta izvajanja projekta.

Kar pravzaprav želimo doseči s projektom IT, ni stvar in odgovornost informatikov, ampak vseh udeležencev, ki sodelujejo v projektu. Napredek v projektih IT kot poslovnih projektih se meri z doseganjem zastavljenih mejnikov na rok v okviru proračuna ter s kakovostjo projektnih pridobitev, ki naj bi zagotavljali poslovne učinke in koristi, do katerih pride šele v obdobju po zaključku projekta. In še bolj kot projektna tveganja so pomembna poslovna tveganja, ki nastopijo z začetkom uporabe rezultatov projektov IT (Štempihar, 2010, str. 8–10).

Namen magistrskega dela je preučevanje managementa kakovosti razvoja programske opreme s teoretičnega vidika. S pomočjo strokovne literature želim preučiti, ali sta izvajanje in management kakovosti pri razvoju programske opreme pomembna.

Magistrsko delo bo sestavljeno iz sedmih glavnih poglavij, ki predstavljajo temelj za management

kakovosti. V uvodnem poglavju bo obrazloženo in definirano, kaj kakovost sploh je ter kaj pomeni management kakovosti. V nadaljnjih treh poglavjih bodo natančno obravnavani vsi trije procesi kakovosti, tj. planiranje, zagotavljanje in kontrola. Posebej bo v poglavju kontrole poudarek na testiranju programske opreme, ki je glavni način preverjanja kakovosti. Nato bodo predstavljeni še nekateri najbolj znani modeli zrelosti. Nadaljuje se z najpomembnejšimi standardi kakovosti v IT oz. v razvoju programske opreme, ki so nujno potrebni, če želimo svoj produkt dati na tržišče. Tržišče predstavljajo danes uporabniki, ki imajo različne želje, pričakovanja, potrebe in zahteve. Največji del potrošnikov lahko tako zajamemo z izpolnjevanjem standardov, saj jim potrošniki najbolj zaupajo, če izdelka še ne poznajo. Magistrsko delo se zaključi s primerom iz prakse.

Magistrsko delo temelji na poglobljenem teoretičnem in analitičnem pregledu strokovne literature, znanstvenih člankov, raziskav, člankov strokovnjakov s preučevanega področja, uporabi slovarjev in zakonov, ki se nanašajo na obravnavano temo. Uporabljene so splošne teoretične metode, kot so: deskriptivna metoda, s katero se opisuje dejstva, procese in pojme; komparativna metoda, s katero se primerja različna dejstva, mnenja in odnose; zgodovinska metoda, s katero se obravnava, kako so se različni postopki, procesi, teorije in podobno razvijali do danes; metoda analize in sinteze, s katero se primerja, razčlenjuje in združuje različna mnenja avtorjev, povezuje različne dele ter obravnava posamezne dele glede na celoto; metoda abstrakcije in konkretizacije, s katero se poudari posamezne lastnosti preučevane teme in konkretizira, kako naj potekajo, se izvajajo in uporabijo obravnavani procesi in aktivnosti; in metodi generalizacije in specializacije, ki omogočata sklepanje od posameznih dejstev na splošno in dvigovanje pojmov nižjega reda na višjega ter oblikovanje specifičnih pojmov in posamično sklepanje. Za uporabo navedenih metod so uporabljeni različni viri, ki jih lahko razdelimo na primarne, sekundarne in terciarne. V primarnih so zajeti strokovni članki, zakonodaja, magistrska in diplomska dela, znanstveni članki in knjige, medtem ko so v sekundarnih virih zajeti učbeniki, študijsko gradivo, slovarji in podobno. Terciarni viri vsebujejo informacije iz primarnih in sekundarnih virov.

1 MANAGEMENT KAKOVOSTI

Danes je kakovost produkta zelo pomembna, saj se ponudniki navadno razlikujejo ravno po kakovosti. Zaradi vse večje konkurence in pritiskov na trgu vsi težijo k zagotavljanju čim boljše kakovosti. Za programsko opremo in produkte informacijske tehnologije je kakovost eden od pomembnejših ciljev, ker z njo podjetje zagotavlja nemoteno delovanje posameznih uporabnikov. Če produkt ni kakovosten, prihaja do velikega števila hroščev (angl. *bugs*), katerih odprava je lahko zelo draga in zamudna, večkrat pa je potrebno tudi ponovno opraviti že opravljeno delo (angl. *rework*). Posledica tega je lahko prekoračitev zastavljenega časa za dokončanje produkta, prekoračitev planiranih stroškov, neustreznost produkta itd. Podjetja zato preizkusijo delovanje programske opreme in jo razhroščujejo (angl. *debugging*), da bi zagotovili ustrezno kakovost.

Različni avtorji definirajo kakovost na različne načine. Reeves in Bednar (1994, str. 419–445) menita, da je kakovost v podjetju povezana s štirimi opredelitvami. V ozadju le-teh lahko najdemo zgodovinske korenine (Burton Swanson, 1997, str. 845–850):

- **Kakovost kot odličnost** sega že v čas grške filozofije in njihovih idealov, in to predvsem na področju človeških dosežkov. V grških časih je to predstavljalo delo najboljših umetnikov, medtem ko danes predstavlja predvsem podlago za oglaševanje nekaterih industrijskih proizvodov, kot so npr.: avtomobili Mercedes.
- **Kakovost kot vrednost** izhaja iz 18. stoletja, ko sta poslovanje in tržišče avtoritativna razsodnika kakovosti, ki se odraža. V tem času je veljalo, da ima dobro blago tudi primerno ceno. Danes to prepoznamo pri trgovskem oglaševanju večjih IT podjetij, kot so npr. Apple, HP, Oracle itd.
- **Kakovost je skladna s specifikacijami** izhaja iz 19. stoletja, ko se je pojavila masovna proizvodnja in zahteva po zamenljivosti posameznih delov. Danes je to sodobni zasuk na področju oglaševanja računalništva, ki združuje odprtost arhitekture sistema in delovanje na vseh platformah.
- **Kakovost je srečanje in/ali preseganje pričakovanj kupcev;** gre za sodoben pojem na področju trženja in nenadni dvig obsega storitvenih dejavnosti. Danes se to odraža v oglaševanju, ki potrošniku nudi zadoščenje, zadovoljstvo in veselje ob nakupu oglaševanega produkta. Dober primer je nova programska oprema, kjer potrošniku obljubljajo nove možnosti, enostavnost uporabe, najboljšo pomoč in podporo uporabnikom.

Avtorja Reeves in Badnar (1994, str. 419–445) opozarjata, da ima vsaka definicija kakovosti svoje prednosti in slabosti za podjetja. S tem sta želela poudariti, da ni mogoče vsega izvajati tako, kot zapišemo, na primer noben produkt nima 100 % kakovosti. Vsaka programska oprema ima hrošče, ki se pojavijo in jih je potrebno odpraviti (Schwalbe, 2009, str. 292).

Kakovost je v Slovarju slovenskega knjižnega jezika definirana kot nekaj »kar opredeljuje kaj glede na veliko mero pozitivnih lastnosti«. Kakovosti z drugo besedo pravimo tudi kvaliteta (Kakovost, 2011).

Mednarodna organizacija za standardizacijo (angl. International Organisation for Standardisation – ISO) definira kakovost kot »skupek lastnosti entitete, ki ima sposobnost, da izpolnjuje določene ali neizražene potrebe (ISO 8042: 1994)« ali kot »stopnjo, do katere niz povezanih značilnosti izpolnjuje zahteve (ISO 9000: 2000)« (Ho & Fung, 2008).

Kousholt (2007, str. 154–162, 225–230, 307–313) ocenjuje kakovost produkta ali storitve kot kaj uporabnik/kupec dobi. Pri tem poudarja, da je doseganje kakovosti pogojeno z uporabnikovimi potrebami, željami, zahtevami in pričakovanji. Ko je vse to doseženo, je produkt kakovosten. Kakovost pomeni tudi definiranje stopnje nepopolnosti, skladnost s specifikacijami, primernost za uporabo, skladnost z zahtevami in odraža celotno delovanje podjetja na področju kakovosti. Kakovost in lastnosti produkta ali storitve predstavljajo celoto, ki je povezana in ima sposobnost zagotavljanja izpolnitve določenih ali neizraženih potreb. Poudarja, da le-ta vsebuje tudi neizražene potrebe, ki so zelo pomembne in se jih zato ne sme zanemariti.

Goncalves, Varejao, Martinho in Cruz (2010, str. 214–219) kakovost definirajo kot večdimenzionalen koncept, ki mora biti skrbno in previdno načrtovan in izpeljan. Pri tem dodaja, da

kakovost ne sme biti omejena na aktivnosti po organizacijskih stopnjah, ampak mora biti vključena v celoten proces izvajanja.

Drugi strokovnjaki kakovost definirajo na osnovi skladnosti z zahtevami in s primernostjo za uporabo. **Skladnost z zahtevami** pomeni, da so procesi projekta in produkt dosegli zapisane specifikacije. Če želimo to doseči, moramo dobro zastaviti obseg projekta in splanirati njegov potek. Če se proces zalomi, bo posledica velikokrat neskladnost produkta z navedenimi zahtevami naročnika. **Primernost za uporabo** pomeni, da je produkt lahko uporabljen, kot je bilo zamišljeno, predvideno. Če produkt tega ne izpolnjuje, ga naročnik in bodoči uporabnik ne moreta uporabljati. Drugače povedano: programska oprema, ki jo to doleti, je neuporabna za naročnika oz. končnega uporabnika (Schwalbe, 2009, str. 294–295).

Velikokrat povezujemo kakovost s pričakovanji kupcev, ki pa posamezen produkt ali storitev vrednotijo subjektivno. Posameznik povprašuje po produktu, ki najbolj ustreza njegovim željam, pričakovanjem, potrebam itd. Kupec tako določa, kaj je zanj kakovostno. Na ravni podjetja ni nihanja kakovosti, saj je le-ta praviloma natančno definirana, skrbno načrtovana in izvedena v skladu z zastavljenimi standardi. Standardi kakovosti podjetjem pomagajo pri določanju okvirja, znotraj katerega zagotavljajo želeno kakovost. S standardi lahko tudi iščemo rešitve za trenutno slabo zagotavljanje kakovosti; to dosežemo s preventivnimi in korektivnimi ukrepi.

Field in Keller (2007, str. 311–325) ocenjujeta, da se preveč definicij sklicuje dobesedno na pričakovanja namesto na zahteve. Pravita, da se problem pojavlja ob izpolnjevanju pričakovanj, ker običajno niso eksplicitno definirana. Nadalje navezujeta, da le-te ne morejo biti spremenjene. Zahteve, definirane v začetni fazi projekta, običajno niso zahteve naročnika, ki jih želi na koncu projekta.

Kakovost posameznega produkta preverjamo tako, da produkt testiramo v čim več fazah procesa izdelave. Končni rezultati testiranja pokažejo, ali smo dosegli želeno kakovost in ali so potrebne dodatne izboljšave, popravki, odprave večjih napak. Moramo tudi vedeti, da je kakovost pri programski opremi vedno nepopolna. Do tega pride, ker testerji, programerji in vsi ostali udeleženci pri izvedbi in testiranju produkta ne morejo predvideti vseh načinov uporabe. Z novim načinom uporabe tako lahko najdemo dodatne napake, hrošče, neizpolnjena pričakovanja in podobno. Zato je zelo pomembno, da pri razvoju nove programske opreme tesno sodelujemo z naročnikom in sproti preverjamo, če smo prav razumeli njegove zahteve, želje, pričakovanja.

Produkt doseže želeno kakovost in primernost, ko je tak, kot si ga je kupec ali stranka zamislil in lahko služi namenu, za katerega je bil razvit. Ob tem je višina kakovosti določena glede na število težav, ki se pojavljajo, ter napak, ki jih še vedno ima.

Jasni opredelitvi, kaj je kakovost sedaj, lahko sledi tudi opredelitev managementa kakovosti (angl. *quality management*). Največkrat se ta pojavlja v obliki projektov na področju informacijske tehnologije, zato bom v nadaljevanju govorila o projektnem managementu kakovosti. Tako vsak

produkt predstavlja nek projekt, ki ga podjetje razvija.

Projektni management kakovosti vključuje procese in aktivnosti podjetja, ki projekt izvaja. Ti določajo politike kakovosti, cilje in odgovornosti, da bo projekt zadostil potrebe, ki so mu bile ob začetku določene. Sistem vodenja kakovosti se izvaja s pomočjo politik podjetja in postopkov; te spremljajo stalne izboljšave procesa, ki se izkažejo za primerne (PMI, 2008, str. 189–190).

Projektni management kakovosti obravnava management projekta in produkta. Nanaša se na vse projekte ne glede na naravo njihovega produkta. Kakovost produkta ima različna merila in tehnike glede na specifičen tip produkta, razvitega v okviru projekta. Vodenje in zagotavljanje kakovosti produktov programske opreme uporabljata različne pristope in merila od ostalih produktov, ki niso vezani na panogo informacijske tehnologije. Pri projektnem managementu kakovosti lahko uporabimo enake pristope za različne panoge. V vsakem primeru je pomembno, da dosežemo zahteve kakovosti produkta ali projekta, saj se v nasprotnem primeru srečujejo z negativnimi posledicami vsi udeleženci, tudi naročnik oz. stranka (PMI, 2008, str. 189–190).

Namen managementa kakovosti je zagotoviti, da bo projekt izpolnil potrebe, za katere je bil tudi vzpostavljen. Poudariti je potrebno, da projektni management vključuje zadoščenje ali preseganje potreb in pričakovanj naročnika oziroma stranke. Projektni tim mora razviti dobre odnose s ključnim naročnikom (angl. *key stakeholder*), še posebej z uporabnikom razvijajočega produkta, da lahko razumejo, kaj za njiju pomeni kakovost. Končno se uporabnik in naročnik kasneje tudi odločita, ali je kakovost razvitega produkta sprejemljiva (angl. *quality acceptance*). Kakovost naj bo pri vsem navedenem na istem nivoju, kot so obseg projekta, ocenjen čas in stroški. Če dosegamo samo zapisane zahteve glede obsega, časa in stroškov projekta, se moramo zavedati, da to ni dovolj. Da je končni produkt ustrezen, mora projektni tim upoštevati tudi posredno izražene želje, potrebe in zahteve (Schwalbe, 2009, str. 295).

Management kakovosti določa načela in metodološki okvir za poslovanje in tako usklajuje dejavnosti za upravljanje in nadzor podjetja v zvezi s kakovostjo. Zagotavljanje in kontrola kakovosti sta dela vsakega uspešnega managementa kakovosti (CIMO guide, 2008, str. 1–18).

Moderni management kakovosti dopolnjuje projektni management in poudarja pomembnost naslednjih področij (PMI, 2008, str. 190–191):

- **zadovoljstvo strank.** Obsega razumevanje, ocenjevanje, definiranje in upravljanje pričakovanj, da so zahteve naročnika dosežene. Zahteve so kombinacija skladnosti z zahtevami in primernostjo uporabe.
- **preprečevanje namesto pregleda.** Eno od temeljnih načel modernega managementa kakovosti navaja, da je kakovost planirana, zasnovana in zgrajena. Stroški preprečevanja napak so bistveno nižji od stroškov odpravljanja in popravljanja napak, ki jih najdemo s pomočjo pregledovanja.

- **stalne izboljšave.** Načrtovati – narediti – preveriti – udejanjiti je osnova za cikel izboljšav kakovosti, ki jih je definiral Shewart in prilagodil Deming. Izboljšave kakovosti bi se morale poznati tako na managementu projekta kot tudi na samem produktu projekta.
- **management odgovornosti.** Uspeh zahteva sodelovanje vseh članov projektnega tima, vendar ostaja zagotavljanje potrebnih virov za uspeh v pristojnosti upravitelja.

Management kakovosti projekta obsega postopke in aktivnosti, ki jih je potrebno opraviti znotraj organizacije, da bi določili politike, cilje in odgovornosti, da bo projekt ustrezno izveden. Politike in postopki se izvajajo ves čas trajanja projekta. Vključuje se tri glavne procese (PMI, 2008, str. 189, 192, 201–202, 206–207; Schwalbe, 2009, str. 295–296):

- **Planiranje kakovosti**
  Vključuje prepoznavanje, kateri standardi kakovosti so pomembni za projekt in kako te standarde izpolnjevati. Vključitev standardov kakovosti v zasnovo projekta je ključni del planiranja. Glavni rezultati planiranja kakovosti so: načrt vodenja in upravljanja kakovosti, kontrolni seznam kakovosti, načrt izboljšav procesov, posodobitve dokumentov projekta, določitev načina za dokazovanje ustrezne kakovosti in metrika kakovosti (metrika je standard ali mera).
- **Zagotavljanje kakovosti**
  Vključuje redno ovrednotenje celotne zmogljivosti za izvedbo projekta z namenom, da zagotovi zadovoljitev pomembnih standardov kakovosti v okviru projekta. Zagotavljanje kakovosti je proces, ki vključuje prevzemanje odgovornosti za kakovost projekta skozi njegov življenjski cikel. Glavni rezultati zagotavljanja kakovosti so: posodobitev sredstev procesa podjetja, sprememba zahtev, posodobitev načrta projektnega managementa in posodobitev dokumentov projekta.
- **Kontroliranje kakovosti**
  Zajema spremljanje in kontroliranje specifičnih projektnih rezultatov za zagotavljanje njihove skladnosti z ustreznimi standardi kakovosti. Pri tem poskušamo najti način, kako izboljšati celotno kakovost. Glavni rezultati kontroliranja kakovosti so: kontrola merjenja kakovosti, potrditev sprememb, potrditev rezultatov, posodobitev sredstev procesa podjetja, sprememba zahtev, posodobitev načrta projektnega managementa in posodobitev dokumentov projekta.

2 PLANIRANJE KAKOVOSTI

Project Management Institute (2008, str. 192–201) planiranje kakovosti definira kot proces opredeljevanja zahtev kakovosti in standardov, ki veljajo za projekt, produkt, njihovo dokumentiranje in določitev načina za dokazovanje skladnosti kakovosti. Planiranje kakovosti je potrebno že v fazi planiranja izvajati vzporedno z drugimi procesi. V tem primeru bo izdelek lahko izpolnil pričakovane standarde kakovosti, njegova izvedba pa bo ostala v okviru pričakovanih stroškov in terminskega plana izvedbe.

Prvi korak k zagotovitvi kakovosti je njeno planiranje. Načrtovanje kakovosti informacijskih projektov pomeni zmožnost predvidevanja situacij in priprava ustreznih ukrepov z namenom

doseganja želenih rezultatov. Ukrepi morajo biti definirani tako, da so razumljivi in popolni. Moderni management kakovosti se osredotoča na preprečevanje napak s programi za izbiro ustreznega materiala, z usposabljanjem in izobraževanjem ljudi o kakovosti ter z načrtovanjem procesov, ki zagotavljajo ustrezne rezultate. Pri planiranju kakovosti je pomembno, da opredelimo ustrezne standarde kakovosti, kot so npr.: ISO standardi in opišemo vse pomembne faktorje, ki neposredno prispevajo k razumevanju zahtev naročnika oz. kupca. Opis produkta, cilji in obseg projekta ter z njimi povezane regulative in standardi so eden najpomembnejših prispevkov procesa načrtovanja kakovosti (Schwalbe, 2009, str. 296–298).

Bøegh, Depanfilis, Kitchenham in Pasquini (1999, str. 68–77) so prepričani, da če želimo narediti dober načrt kakovosti na področju programske opreme, moramo **kakovost** najprej **specificirati**. To pomeni vzpostaviti zahteve kakovosti za produkt programske opreme, v katero so vključene tri glavne aktivnosti: izbira modela kakovosti, izvedba analize izvedljivosti in prepoznavanje delov produkta, ciljev ter zahtev. Specifikacija kakovosti je pomembna, ker se lahko razlikuje med posameznimi deli razvoja programske opreme. Ko je to definirano, lahko kakovost začnemo načrtovati, kar vključuje odločanje za primeren razvojni proces in določanje ciljne vrednosti za kakovost notranjih atributov. Potrebno je **izbrati razvojni proces** za vsak del produkta, ki ga želimo razviti. Razvojni proces naj bo osnovan na podlagi zahtevanih lastnosti in preteklih izkušenj podjetja s podobnimi projekti. Avtorji opozarjajo, da je ta del zelo pomemben, saj ga mora projektni tim vključiti v svoj razvoj produkta tako, da le-ta zajema vse zahteve. **Določanje ciljne vrednosti za kakovost notranjih lastnosti** je opredeljeno v modelu kakovosti. Kakovost notranjih lastnosti je definirana v obliki merljivih ciljev. Lastnosti so povezane z vmesnimi dejavnostmi in rezultati, na podlagi katerih se bo napredek tudi spremljal in nadzoroval. Če se podjetje še ni srečalo s podobnim definiranjem ciljnih vrednosti notranjih lastnosti in ciljev, avtorji priporočajo posvet s strokovnjaki s tega področja.

Sommerville (2004) trdi, da načrt kakovosti določa kakovost želenega produkta, kako se bo le-te ocenjevalo in definira ter opredeljuje kakovost najpomembnejših lastnosti. Načrt mora poleg tega določiti postopek za ocenjevanje kakovosti, ki temelji na standardih organizacije in na novo vpeljanih standardih, če so le-ti potrebni. Struktura plana kakovosti naj bo sledeča: uvod o produktu, načrti produkta, opis procesov izvedbe, cilji kakovosti, tveganja in upravljanje s tveganji. Ta dokument mora biti kratek in jedrnat, saj ga v nasprotnem primeru nihče ne bo bral, kar posledično vodi v neuspešnost.

Avstralski strokovnjaki menijo, da priprava načrta kakovosti ni kompleksna. Načrt vključuje opredelitev vseh pričakovanih rezultatov na začetku projekta in odločitev, kako bomo najbolje preverili njihovo kakovost. Današnja ustaljena praksa je prekomerno sprotno preverjanje kakovosti, da bi tako preprečili ali vsaj omejili težave z napakami v kasnejših fazah razvoja programske opreme. Zavedati se je potrebno, da čim kasneje odkrijemo napako, tem težje jo odpravimo, kar posledično pomeni tudi več porabljenega časa in več stroškov za projekt. Potrebno je predvideti tudi mehanizem za odpravo napak, ki bo zagotovil postopek za njihovo odpravljanje, ter upoštevanje dodatnega časa pri planiranju za njihovo odpravljanje v fazi razvoja produkta. Poudarjajo tudi, da je

naročnik veliko mirnejši in zadovoljnejši, če vidi, da produkt dosega predvideno kakovost že v času izvajanja projekta (Project Quality Planning, 2004).

Drugi strokovnjaki so mnenja, da je za visoko kakovostne proizvode potrebno imeti strateški plan kakovosti. Prisotnost strateškega načrtovanja se odraža v smernicah vizije, poslanstva in politike kakovosti podjetja, ki jih skupaj imenujemo tudi »izjave o kakovosti (angl. *quality statements*)«. Bistvena ideja strateškega načrtovanja kakovosti je v vrednosti za kupca. Izdelek mora imeti poudarek na vrednosti za kupca namesto na fizičnih lastnostih. Tega ni mogoče doseči, če podjetje nima prisotne »kulture kakovosti (angl. *culture of quality*)«, strategije in plana za izvedbo. Strateško planiranje kakovosti poteka v sedmih korakih, ki so: odkrivanje prihodnjih potreb kupcev, pozicioniranje kupcev, predvidevanje prihodnosti, analiza vrzeli med trenutnim in prihodnjim stanjem, zapolnjevanje vrzeli iz prejšnjega koraka, usklajenost načrta z izjavami o kakovosti in izvedba načrta, ki je najtežja (Strategic Quality Planning, 2008).

Pri informacijskih projektih pogosto težko popolnoma razumemo strankine želje glede zmogljivosti produkta, saj se tehnologija – strojna in programska oprema ter mrežne tehnologije ves čas spreminjajo. Pomemben vidik informacijskih projektov, ki vplivajo na kakovost projekta, vključuje funkcionalnost in značilnosti, sistemske izhode, zmogljivost, zanesljivost in vzdrževanje, ki jih bom v nadaljevanju tudi podrobneje razložila (Schwalbe, 2009, str. 296–298; Schwalbe, 2007, str. 309–312):

- **Funkcionalnost (angl. *Functionality*)** pomeni stopnjo, do katere sistem opravlja svoje predvidene funkcije. Funkcije se nanašajo na značilnosti sistema, kot jih vidijo uporabniki. Potrebno je razjasniti, katere funkcije in funkcionalnosti so nujno potrebne in katere so le možnosti. Obvezna funkcionalnost je na primer omogočiti uporabniku spremljanje prodaje po produktih, dodatna možnost pa je grafični uporabniški vmesnik z uporabo ikon, menijev itd.
- **Sistemski izhodi (angl. *System outputs*)** so zajete zaslonske slike in poročila, ki jih ustvari sistem. Zelo pomembno je, da natančno določimo in definiramo, kako naj zajete zaslonske slike in poročila izgledajo za sistem. Ti morajo uporabnikom zagotoviti enostavno interpretacijo in primeren format izhodov.
- **Zmogljivost (angl. *Performance*)** je način, kako dobro proizvod ali storitev služi namenu uporabe naročnika. Pri načrtovanju sistema visoke zmogljivost se udeleženci projekta srečujejo z veliko problemi, saj morajo odgovoriti na številna vprašanja v povezavi z obdelovanjem podatkov, z določanjem obdelovanih podatkov, s številom sočasnih dostopov in njihovo prihodnje naraščanje, z zmogljivostjo računalniške opreme, na kateri bo programska oprema delovala, s hitrostjo odzivnosti itd. V informacijskem sistemu organizacije se na problem zmogljivosti nanašajo naslednje težave: izpad sistema večkrat v mesecu, nezadovoljstvo uporabnikov z odzivnim časom sistema itd. Nekatere navedene težave se lahko odpravi zgolj z nadgradnjami, druge pa so posledice neustrezne kakovosti, katerih odprava je kompleksna in draga, še posebej, če je sistem že v uporabi.
- **Zanesljivost (angl. *Reliability*)** je sposobnost sistema, da v normalnih pogojih pravilno deluje.
- **Možnost vzdrževanja (angl. *Maintainability*)** se nanaša na vzdrževanje produkta. Cilj vzdrževanja je táko vodenje sistema, da bo le-ta deloval in bo odziven, ko bo šel v uporabo

(Burton Swanson, 1997, str. 845–850). Večina informacijskih projektov ne zagotavlja stoodstotne zanesljivosti, zato mora naročnik določili svoja pričakovanja. V vzdrževanje informacijskega sistema podjetja je vključeno tudi nalaganje novih podatkov v sistem in izvajanje vzdrževalnih postopkov na strojni in programski opremi. Pri tem je dobro doreči, kakšne so posledice za uporabnika v času nadgrajevanja ali odpravljanja napak ter sprejemljivost posledic vzdrževanja.

Zgoraj je naštetih le nekaj zahtev, ki se nanašajo na načrtovanje kakovosti. Projektni managerji morajo s člani projektnega tima razmisliti o vseh vidikih in določiti cilje za zagotavljanje kakovosti. Svoje vloge se mora zavedati tudi naročnik, katerega dolžnost je navesti kritične potrebe kakovosti. Vsi udeleženci projekta morajo sodelovati tako, da se dogovorijo in skupno poiščejo ravnotežje med kakovostjo, obsegom, časom in stroški projekta. Projektni manager mora biti seznanjen z osnovno terminologijo, standardi in viri, ki veljajo za področje managementa kakovosti, hkrati pa je tudi odgovoren za končno kakovost produkta (Schwalbe, 2009, str. 298).

2.1 Tehnike in orodja za planiranje kakovosti

Za planiranje kakovosti se uporabljajo različne tehnike in orodja. V nadaljevanju bom predstavila najpogosteje uporabljene. Poleg teh obstaja še veliko drugih, ki se jih uporablja pri različnih tipih projektov ali na različnih področjih uporabe.

2.1.1 Analiza stroškov in koristi (angl. *Cost-Benefit Analysis*)

Glavne prednosti dosežene zahtevane kakovosti so manjša potreba po ponovnem opravljanju dela (angl. rework), višja produktivnost, nižji stroški in povečano zadovoljstvo naročnikov. Z analizo stroškov in koristi primerjamo vsako aktivnost kakovosti s stroški nastalimi zaradi kakovosti in koristmi, ki smo jih pridobili (Quality Planning, 2011; PMI, 2008, str. 195).

2.1.2 Stroški kakovosti (angl. *Cost of Quality – COQ*)

Vključujejo vse stroške, ki nastanejo v življenjskem ciklu produkta, s katerimi se preprečuje slaba kakovost izdelka in neskladnost s podanimi zahtevami. Dejansko so to stroški ustreznosti plus stroški neustreznosti produkta. Če je produkt ustrezen, pomeni, da je zadostil vsem zahtevam in je tudi primeren za uporabo. V primeru neustreznosti produkta nismo zadostili pričakovani kakovosti in moramo za napake tudi prevzeti odgovornost, tedaj govorimo o stroških neuspeha oz. o stroških slabe kakovosti (PMI, 2008, str. 195; Schwalbe, 2009, str. 297).

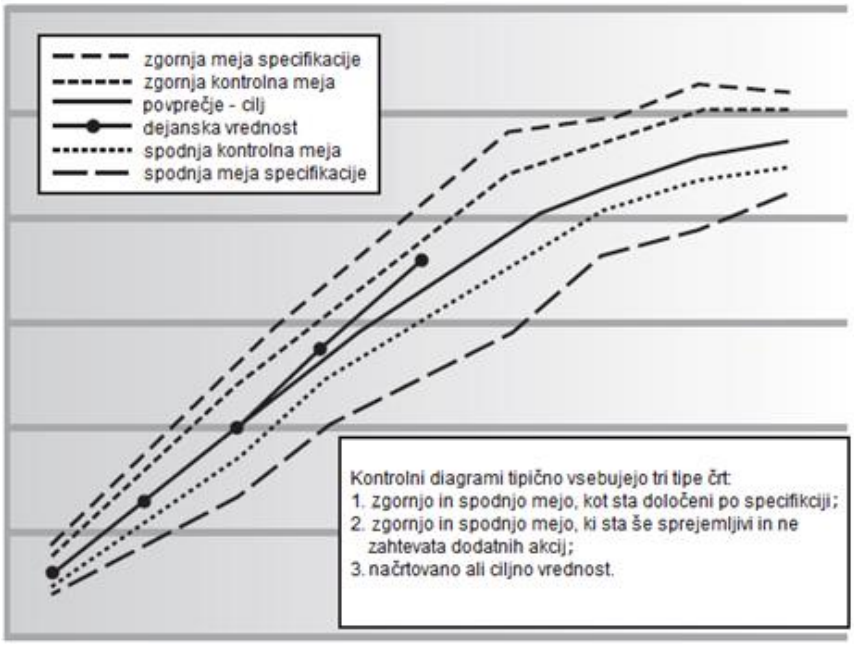
Stroški preprečevanja neuspeha projekta so stroški planiranja in izvajanja projekta, ko napak nimamo ali so le-te znotraj sprejemljivega ranga. To so stroški izobraževanj, urejanja, dodatne opreme, procesov dokumentacije ipd. **Stroški preverjanja ustreznosti** kakovosti so tisti, s katerimi ovrednotimo procese in njihove izhode, da zagotovimo projekt brez napak ali so le-te znotraj sprejemljivega ranga. To so stroški testiranja, pregledovanja, pregledov vzdrževanja, testiranja opreme in uničujoče izgube testiranja (Quality Planning, 2011; PMI, 2008, str. 195; Cost of Quality, 2011).

Stroške neuspeha delimo na notranje in zunanje. **Notranji** so tisti, ki so nastali znotraj projekta in smo jih odkrili, preden je naročnik prejel produkt. Stroški notranjega neuspeha so ponovno delo; to so stroški povezani z zamudo pri plačilu računov, pri popisu stroškov, ki so neposredna posledica napake, pri stroških prezgodnje odpovedi produkta itd. **Zunanji** stroški neuspeha so povezani z napakami, ki niso bile odkrite in popravljene, preden smo naročniku predali produkt. Ti predstavljajo odgovornost za neustrezni produkt, garancijsko delo in stroške, obravnavanje pritožb, stroške usposabljanja servisnega osebja, prihodnja izguba poslovanja itd. (Quality Planning, 2011; PMI, 2008, str. 195; Cost of Quality, 2008).

2.1.3 Kontrolni diagram (angl. *Control Charts*)

Kontrolni diagram je grafični prikaz podatkov, ki prikazujejo rezultate procesa skozi čas. Uporabljajo se za ugotavljanje, ali je proces stabilen in ali napreduje skladno s pričakovanji. Zgornja in spodnja meja v diagramu sta določeni na podlagi zahtev pogodbe in odražata največje in najmanjše dovoljene vrednosti. Prestop dovoljenih mej je lahko kaznovan tudi s penali in je zanj potrebno najti tudi razlog, zakaj se je zgodil. Zgornjo in spodnjo mejo nadzora določijo manager projekta in zainteresirane strani (navadno naročnik) na način, da določijo točke, kjer bo potrebno sprožiti korektivne ukrepe za preprečitev odstopanj. Za ponavljajoče se procese je meja nadzora definirana kot ± 3σ (standardne odklone). Določen postopek se šteje, da je ušel izpod nadzora, ko je podatkovna točka presegla mejne kontrole ali ko je sedem zaporednih točk nad ali pod povprečjem (Data Collection and Analysis Tools, 2011; Basic Tools for Process Improvement – Moule 10: Control Chart, 2011; Jarrett, 1995, str. 333–339; Schwalbe, 2009, str. 301–302).

Slika 1: Primer kontrolnega diagrama.



Vir: PMI, PMBOK, 2008, str. 196.

Kontrolni diagrami se uporabljajo za spremljanje različnih tipov izhodnih spremenljivk. Najpogosteje ga uporabljamo za sledenje ponavljajočih se aktivnosti, kot je masovna proizvodnja.

Uporabljamo jih lahko tudi za nadzor stroškov in odstopanj časa, obsega in pogostostih sprememb ali druge rezultate upravljanja z namenom ugotavljanja, ali je potek projekta pod nadzorom. Slika 1 prikazuje kontrolni diagram, ki sledi porabljenemu času za projekt (Basic Tools for Process Improvement – Moule 10: Control Chart,, 2011; PMI, 2008, str. 196–197; Control Charts, 2011).

2.1.4 Izdelava primerjalnih analiz (angl. *Benchmarking*)

Primerjalna analiza vključuje primerjavo dejanskih ali načrtovanih projektov s primerljivimi projekti z namenom ugotavljanja najboljših praks, iskanja idej za izboljšanje in zagotavljanja podlage za merjenje uspešnosti. Projekti, ki so uporabljeni za primerjavo, lahko izhajajo iz podjetja ali zunaj njega ter z enakega ali drugačnega področja (PMI, 2008, str. 197; Schwalbe, 2009, str. 299).

Rezultati uporabe tega orodja so vidni v možnostih postavljanja novih ciljev in boljšega poslovanja podjetja. Aktivnosti procesa so vidne v zadovoljstvu kupcev, uporabnikov in naročnikov, kar dosežemo s ponudbo kakovostnejših produktov na tržišču. Primerjalna analiza je neprekinjen proces, ki zahteva spreminjajoči se trg, tehnologijo in širše okolje (Sidoroska Ćorić, 2005, str. 15).

2.1.5 Načrtovanje eksperimentov *(*angl. *Design of Experiments - DOE)*

Načrtovanje eksperimentov je tehnika planiranja kakovosti, ki pomaga določiti, katera spremenljivka ima največji vpliv pri končnem produktu posameznega procesa (Schwalbe, 2009, str. 296–297). Je tudi statistična metoda za ugotavljanje dejavnikov, ki lahko vplivajo na določene spremenljivke produkta ali postopka v razvoju ali v proizvodnji. V fazi načrtovanja kakovosti se uporablja za določitev števila in vrste testov ter njihov vpliv na stroške kakovosti (PMI, 2008, str. 197–198).

2.1.6 Statistično vzorčenje (angl. *Statistical Sampling*)

Statistično vzorčenje je ključni koncept managementa kakovosti. Vključuje izbor dela populacije z interesnim področjem za pregled (Schwalbe, 2009, str. 306). Frekvenca vzorčenja se določi med procesom planiranja kakovosti. Tako stroški vključujejo na primer število testov in pričakovan odpad. Statistično vzorčenje je zelo obsežno področje. Na nekaterih področjih uporabe je potrebno projektni tim seznaniti z različnimi tehnikami vzorčenja, da lahko izbere ustrezen vzorec, ki dejansko predstavlja populacijo z interesnim področjem (PMI, 2008, str. 198).

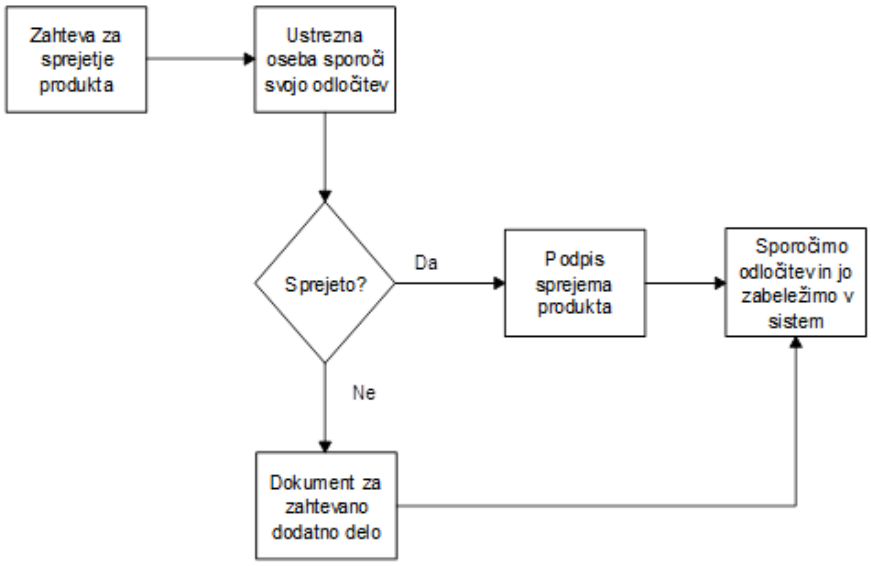
2.1.7 Diagram poteka (angl. *Flowcharting, Flowcharts*)

Diagram poteka je grafični prikaz logike in poteka procesa. Pomaga pri analiziranju, kako se je problem pojavil in kako lahko proces še izboljšamo. Diagrami poteka prikazujejo potek aktivnosti, točke odločitve in vrstni red v procesu (Schwalbe, 2009, str. 305). Pri pripravi diagrama uporabljamo različne simbole. Pet najpogosteje uporabljenih je: elipsa (prikazuje začetno in končno točko procesa), pravokotnik (prikazuje posamezni korak ali aktivnost v procesu), romb (predstavlja točko odločitve, t. i. razvejišče; odločamo se med odgovoroma DA in NE, obvezno moramo oba odgovora tudi uporabiti), krog (označuje, da je določen korak povezan z drugim listom ali drugim

delom diagrama poteka) in trikotnik (prikazuje, kje v procesu se pojavijo meritve). Za boljšo predstavo sledi grafični prikaz diagrama poteka na Sliki 2 (Basic Tools for Process Improvement – Module 6: Flowchart, 2011; Flow Charts, 2011).

Diagram poteka lahko pripravimo na treh različnih ravneh natančnosti, in sicer za makro, mini in mikro raven. Makro raven ali t. i. široka slika (angl. *big picture*) je namenjena vodstvu, zato je dovolj, če ima šest prikazanih korakov posameznega procesa. Mini raven spada med mikro in makro raven. Zanjo je značilno, da se osredotoča na del makro ravni. Mikro raven je najnižja raven, lahko bi ji tudi rekli zemeljska raven. Slednja ima zelo podrobno prikazane korake procesa in se jo uporablja za prikaz, kako se določena naloga opravlja. Podrobna upodobitev korakov procesa pomaga udeležencem še dodatno izboljšati izvajanje posameznih korakov (Basic Tools for Process Improvement – Module 6: Flowchart, 2011, 2011).

Slika 2: Primer diagrama poteka



Vir: K. Schwalbe, Information Technology Project Management, 2009, str. 305.

Preučevani diagram najpogosteje uporabljamo za: definiranje in analiziranje procesov, za izdelavo slike procesov »korak-za-korakom« za analize, razpravo ali komunikacijo in za definiranje, standardiziranje ali iskanje področij za izboljšanje obstoječega procesa (Flow Charts, 2011).

V času načrtovanja kakovosti lahko ti diagrami pomagajo projektni skupini preprečiti možne težave s kakovostjo. Zaradi zavedanja morebitnih težav lahko člani projektnega tima razvijejo testne postopke ali pristope za obvladovanje težav (PMI, 2008, str. 189–199).

2.1.8 Metode za zagotavljanje kakovosti, ki so jih razvile različne organizacije (angl. *Proprietary Quality Management Methodologies*)

Za zagotavljanje kakovosti so na voljo različne metodologije, ki jih je mogoče uporabiti že v fazi

načrtovanja, na primer Six Sigma, Lean Six Sigma, CMMI ®, Quality Function Deployment (QFD) itd. (PMI, 2008, str. 199). V svoji nalogi jih bom predstavila v 5. poglavju.

2.1.9 Druga orodja za planiranje kakovosti

Na voljo so tudi druga orodja, ki se pogosto uporabljajo za boljše definiranje zahtev kakovosti in učinkovito načrtovanje kakovosti za posamezno aktivnost, na primer: viharjenje možganov (angl. *Brainstorming*), diagram sorodnosti (angl. *Affinity diagram*), analiza moči in vplivov (angl. *Force field analysis*), tehnika nominalne skupine (ang. *Nominal group technique*), matrični diagrami (angl. *Matrix diagram*) itd. Za planiranje kakovosti je na voljo še več drugih orodij (PMI, 2008, str. 199).

2.2 Rezultat planiranja kakovosti

Rezultat planiranja kakovosti, ki zajema tehnike in orodja ter druge potrebne vhode, kot so obseg projekta (opredelitev meril, obsega, priprava seznama aktivnosti WBS), seznam udeležencev, načrt stroškov, terminski načrt, seznam tveganj itd. lahko strnemo v nekaj dokumentov projektne dokumentacije, kot je predstavljeno v nadaljevanju.

Plan managementa kakovosti opisuje, kako bo projektna skupina v projekt implementirala politiko kakovosti, ki je sestavni del vsakega načrta projekta. Vključuje kontrolo kakovosti, zagotavljanje kakovosti in neprestane izboljšave za projekt. Plan je lahko formalen ali neformalen, zelo podroben ali splošen. Slog in podrobnosti opredeljujejo zahteve projekta. Načrt managementa kakovosti se pregleda kmalu po začetku projekta, da se zagotovi odločitve, ki temeljijo na točnih informacijah. Pav tako je v projektni dokumentaciji potrebno posodobiti register udeležencev pri projektu in matriko odgovornosti (PMI, 2008, str. 200–201).

Metrika kakovosti je operativna opredelitev, ki opisuje lastnosti projekta ali produkta in način, kako bodo merjene. Meritev je dejanska vrednost. Toleranca določa dopustna odstopanja opravljenih meritev. Metrika kakovosti se uporablja za zagotavljanje in nadzor kakovosti procesov. Primeri metrik kakovosti so: pravočasnost izvedbe, nadzor proračuna, pogostost napak, razpoložljivost, zanesljivost in test pokritosti (Quality Planning, 2011).

Kontrolni seznam (angl. *Quality Checklist*) je strukturirano orodje, ki se uporablja za preverjanje, ali so bili opravljeni vsi potrebni koraki. Tako preprosti kot tudi zapleteni kontrolni seznami so osnovani na zahtevah in praksah. Veliko podjetij uporablja standardizirane kontrolne sezname z namenom doslednega in stalnega izvajanja nalog. Seznami kakovosti se uporabljajo v postopku izvajanja nadzora kakovosti (Quality Planning, 2011; PMI, 2008, str. 201).

Načrt izboljšave procesov navaja korake za analizo procesov tako, da se določi tiste aktivnosti, ki jim vrednost narašča. Področja preučevanja so meje procesa (začetek in konec, vhodi in izhodi, potrebni podatki, lastnik, zainteresirane skupine), konfiguracija procesa, metrike, cilji za izboljšavo delovanja (PMI, 2008, str. 201).

3 ZAGOTAVLJANJE KAKOVOSTI

Zagotavljanje kakovosti predstavlja drugo fazo managementa kakovosti. Opravljati jo začnemo že v prvi fazi projekta, vendar mora biti pred njenim začetkom vsaj delno opravljeno načrtovanje kakovosti. Zato tudi Kloppenborg, Patrick in Petrick (2002, str. 1–19) trdijo, da se zagotavljanje kakovosti začne z načrtom, ki je že potrjen s strani ključnih udeležencev projekta. Zaključi se po izboljšavah in dostavi procesov in rezultatov.

Zagotavljanje kakovosti je definirano kot planiran in sistematičen vzorec vseh aktivnosti potrebnih za zagotovitev, da produkt ali del produkta ustreza podanim tehničnim zahtevam. V tem primeru so aktivnosti ustvarjene z namenom ovrednotenja razvojnega procesa posameznega produkta (Galin, 2004, str. 14–34, 56–74; Futrell, Shafer, & Shafer, 2002, str. 1189–1201).

Kenneth H. Rose (2005, str. 61–66) definira zagotavljanje kakovosti kot program kombiniranih zbirk aktivnosti, ki jih izvede projektni tim, da doseže zastavljeni cilj projekta. Aktivnosti opredeli kot zadeve, ki jih projektni tim naredi za izvedbo projekta, da je ta skladen s predvideno kakovostjo in standardi. Pri tej definiciji opozarja, da je definiranje zagotavljanja kakovosti lahko problematično, saj dejansko izvajamo nadziranje nad aktivnostmi. Takahashi (1995, str. 759–765) je mnenja, da je zagotavljanje kakovosti bistvo nadzora kakovosti. Poudarja, da mora biti kakovost vgrajena v vsak proces in se je ne da ustvariti skozi pregledovanje, testiranje.

Schwalbova (2009, str. 298–299) trdi, da zagotavljanje kakovosti vključuje vse aktivnosti, ki so povezane z doseganjem ustreznih standardov kakovosti za projekt. Drugi cilj zagotavljanja kakovosti je neprekinjeno izboljševanje kakovosti. Opaža tudi, da veliko podjetij že razume pomen zagotavljanja kakovosti, zato nekatera v ta namen vzpostavijo tudi posamezne oddelke za obravnavano področje. Da bi zagotovili zahtevano kakovost, imajo podjetja zelo podroben in natančen proces preverjanja kakovosti svojih produktov. Zelo pomembno je ta proces optimizirati, saj morajo biti produkti narejeni po konkurenčni ceni. Največji vpliv na kakovost imajo po njenem mnenju managerji projektov in management podjetja.

PMI (2008, str. 201–202) se podobno kot Schwalbova sklicuje na oddelek zagotavljanja kakovosti oz. organizacije, ki pogosteje nadzirajo aktivnosti zagotavljanja kakovosti. Za razliko od nje pa opredeli še subjekte, ki zagotavljajo podporo kakovosti projekta. K slednjim prištevamo projektni tim, management organizacije, naročnika in sponzorja ter druge udeležence, ki niso aktivno sodelujoči v projektu.

Po tradicionalnem modelu razvoja programske opreme predstavlja zagotavljanje kakovosti niz sistematičnih aktivnosti, ki dokazujejo zmožnost, da skozi zastavljen proces razvoja dosežemo produkt, ki je primeren za predvideno uporabo (Zhao & Elbaum, 2003, str. 65–75).

Aktivnosti zagotavljanja kakovosti vključujejo načrtovan sistem pregledovalnih postopkov. Izvaja jih osebje, ki ni neposredno vključeno v razvojni oz. projektni tim. Najbolje je, če mnenja o kakovosti podajo strokovnjaki s tega področja, ki navadno niso vpleteni v preostali del projekta. S

pregledom postopkov se preverja, ali so bili zastavljeni cilji kakovosti doseženi (IPCC, 2000).

Pri zagotavljanju kakovosti sodelujejo sponzor projekta, zunanja stranka, projektni manager in glavni tim. Sponzor vodi vso komunikacijo z zunanjimi strankami, je mentor projektnemu managerju in rešuje ovire, ko se pojavijo. Zunanja stranka v tem primeru vodi samo zunanje komuniciranje. Projektni manager podobno kot sponzor vodi komunikacijo z zunanjim osebjem, potrjuje ustrezne kvalificirane procese, organizira kakovost, razvija in planira. Tim je zadolžen za uporabo kvalificiranih procesov, iskanje podatkov, iskanje razlogov za pravo smer managementa kakovosti revizije in planiranje nadaljnjega dela (Kloppenborg, Patrick & Petrick, 2002, str. 23–35).

Današnja tendenca hitre rasti razvoja programske opreme postaja vse večji izziv in strošek. Da bi to lahko dosegla, si podjetja želijo čas, stroške, cilje kakovosti, razvojni pristop itd. optimizirati. To pomeni tudi optimizacijo aktivnosti zagotavljanja kakovosti, ki jih razumemo kot vse vrste analitičnih aktivnosti, ki potekajo med razvojem programske opreme z namenom iskanja in odstranjevanja napak. Aktivnosti, ki se osredotočajo na zaznavanje obstoječih napak, imenujemo tudi aktivnosti preverjanja in potrjevanja (angl. *verification and validation activities*). Pomembna strategija v tej smeri je izbira sistematične kombinacije obstoječih tehnik zagotavljanja kakovosti, da bi dosegli nižje stroške ali boljšo učinkovitost (Elberzhager, Münch, & Nha, 2011, str. 1–15).

Za razvoj zagotavljanja kakovosti je potreben točno določen proces. Izvaja se po naslednjih korakih (Kenneth Rose, 2005, str. 62):

- izbira ustreznih standardov oz. specifikacij,
- uporaba definicij, definiranje aktivnosti zbiranja podatkov in primerjanje rezultatov z načrtom. Razvijanje in uporaba metrik,
- definiranje in oskrbovanje z viri,
- pripisovanje odgovornosti specifičnim entitetam,
- sestavljanje aktivnosti v načrtovanje kakovosti.

V nadaljevanju so obravnavane najpogosteje uporabljene tehnike in orodja na področju zagotavljanja kakovosti.

3.1 Tehnike in orodja zagotavljanja kakovosti

Večino tehnik in orodij, ki sem jih omenila že pri planiranju kakovosti, lahko uporabimo tudi pri zagotavljanju kakovosti. Od teh bi najbolj izpostavila izdelavo primerjalnih analiz, ki so zelo pomembne, saj pomagajo najti ideje za izboljšave kakovosti (Schwalbe, 2009, str. 298–299). Del zagotavljanja kakovosti je tudi iskanje stalnih izboljšav. Z vključevanjem izboljšav lahko procese dodatno optimiziramo, jim dvignemo učinkovitost ter kakovost (Elberzhager et al., 2011, str. 1–15). Posledično so naši produkti konkurenčnejši, boljši in imajo možnost doseganja sprejemljive cene glede na ponudbo na tržišču.

3.1.1 Revizija kakovosti (angl. *Quality Audits*)

Ena izmed tehnik, ki je namenjena predvsem zagotavljanju kakovosti, je revizija kakovosti. Predstavlja strukturiran in neodvisen pregled, s katerim ugotovimo, ali so projektne aktivnosti v skladu s politikami, procesi in postopki podjetja in projekta (PMI, 2008, str. 204). Schwalbova (2009, str. 299) navaja, da je revizija kakovosti strukturiran pregled specifičnih aktivnosti managementa kakovosti, ki nam pomaga naučiti se lekcije na področju izboljšav za tekoči projekt in prihodnje. Cilji revizije kakovosti so opredeliti vse dobre in/ali najboljše prakse, ki se izvajajo, opredeliti vse vrzeli in/ali pomanjkljivosti, deliti dobre prakse in jih uvajati oz. izvajati pri podobnih projektih v posamezni organizaciji, industriji, dejavno uporabiti način za izboljšanje izvajanja postopkov pri dvigu produktivnosti in uporabiti prispevek vsake revizije kot eno izmed izkušenj podjetja.

Ena pomembnejših odločitev pri reviziji kakovosti je določitev osebe, ki to opravlja. Strokovnjaki priporočajo, da se za to nalogo določi od projekta ali podjetja čim bolj neodvisno osebo. Če imamo na razpolago dovolj sredstev, naj bo to zunanji izvajalec. Od izbire te osebe je odvisno, kako objektivne bodo ocene. Namen revizije je doseči čim bolj realno mnenje o zagotavljanju in izvajanju kakovosti. Revizorji kakovosti so še posebej dobrodošli v primerih, ko sprejmemo nove metode ocenjevanja ali ko bistveno spremenimo obstoječe metode. Revidiranje kakovosti lahko poteka po točno določenem urniku oz. razporedu ali naključno. Taki pregledi se navadno odvijajo v podjetjih ob koncu določene faze in/ali ob zaključku projekta. Razlike v rezultatih glede na njen čas izvajanja se ne bi smele bistveno razlikovati (IPCC, 2000; Wallance, 2004).

Rezultat revizije kakovosti je dodatno prizadevanje za odpravljanje anomalij, ki lahko vplivajo na zmanjšanje stroškov kakovosti in povečanje sprejemljivosti produkta s strani sponzorja. Namen revizije je tudi spodbuditev udeležencev, da zagotavljajo skladnost s predvideno, začrtano in zahtevano kakovostjo. Največja pomanjkljivost je, da se običajno izvaja prepozno, da bi še lahko vplivala na izvajanje oz. kakovost produkta tekočega projekta. Strokovnjaki priznavajo, da je kljub temu dobra popotnica za projekte, ki sledijo, saj se iz revizije kakovosti lahko veliko naučimo in v prihodnosti tudi pazimo na odkrite pomanjkljivosti (PMI, 2008, str. 204; Wallance, 2004).

3.1.2 Analiza procesov (angl. *Process Analysis*)

Analiza procesov sledi opisanim korakom plana procesa izboljšav. S tem se opredeli potrebne izboljšave. Analiza raziskuje tudi težave in omejitve, na katere smo naleteli, ter aktivnosti, ki ne prinašajo dodane vrednosti ugotovljene skozi delovanje posameznega procesa. Analiza procesov vključuje analizo temeljnih vzrokov, posebne tehnike za identifikacijo problema, vzroke, ki vodijo do teh ter razvoj preventivnih ukrepov (PMI, 2008, str. 204).

3.1.3 Statične in dinamične tehnike zagotavljanja kakovosti (angl. *Static and dynamic techniques*)

Elberzhaget et al. (2011, str. 1–15) se zavedajo, da obstaja mnogo različnih tehnik zagotavljanja kakovosti z namenom doseganja visoke kvalitete končnega produkta. Sami predlagajo sistematično

kombinacijo različnih statičnih in dinamičnih tehnik, ki obljubljajo izkoriščanje sinergijskih učinkov obeh tehnik. Namen izkoriščanja teh učinkov je višja stopnja zaznavanja napak ali zmanjšanje stroškov zagotavljanja kvalitete.

Statične tehnike zagotavljanja kakovosti ne potrebujejo modelov ali kode, da so izvršene. Njihov namen je preučiti predmete, kot so zahtevani dokumenti, oblikovanje modelov ali kode, ki jih ne poženemo. Statične tehnike so lahko na primer pregledovanje, ocenjevanje, »sprehodi čez procese (angl. *walkthroughs*)« ali statistične analize, kot je rezanje programa (angl. *program slicing*) (Elberzhager et al., 2011, str. 1–15).

V nasprotju s statičnimi tehnikami so **dinamične tehnike** zagotavljanja kakovosti, ki morajo izvrševati programe ali dele programov. Te tehnike so lahko analiza mejne vrednosti, nadzor toka na osnovi testiranja tehnik, enakovredno ločevanje ali dinamične analize, kot je profiliranje programov (Elberzhager et al., 2011, str. 1–15).

Uporaba obeh opisanih tehnik za aktivnosti zagotavljanja kakovosti je dobro poznana praksa za področje razvoja programske opreme. Navadno se kombinacija tehnik uporablja na nivoju kode programske opreme. Najpogostejši obliki kombinacije statične in dinamične tehnike predstavljata zbiranje (angl. *compilation*) in povezovanje (angl. *integration*). Pri tem zbiranje pomeni, da se različne statične in dinamične tehnike zagotavljanja kakovosti uporabljajo za skupni cilj. Pomembno je, da se vsaka tehnika odvija v izolaciji, torej brez stika z drugo tehniko. Glavna značilnost povezovanja je, da se tehniki prepletata – in to tako, da je izhod ene tehnike vhod v drugo. Največkrat naletimo, da je uporabljena najprej statična tehnika, ki ji sledi dinamična (izhod statične tehnike je tako vhod dinamične tehnike). Avtorji navajajo, da je večina pristopov, ki vključujejo uporabo statičnih in dinamičnih tehnik, podprtih z orodji ali prototipi orodij, ki jih še razvijajo (Elberzhager et al., 2011, str. 1–15).

Obravnavala sem samo nekaj največkrat uporabljenih tehnik in orodij za zagotavljanje kakovosti. Obstaja še mnogo drugih, ki so predvsem prilagojene na tip projekta in področje delovanja. V nadaljevanju bom predstavila, kaj so rezultati procesa zagotavljanja kakovosti.

3.2 Rezultati zagotavljanja kakovosti

Eden od rezultatov zagotavljanja kakovosti je tudi načrt zagotavljanja kakovosti, ki vključuje: opredelitev produkta in njegovo namembnost, opredelitev kritičnih potreb za razvoj produkta, določitev ciljne publike produkta, utemeljitev projekta, seznam potrebnih stroškov produkta, opis življenjskega cikla uporabljenega modela za razvoja produkta, zajem predvidenih končnih rezultatov produkta, priprava blok diagrama (angl. *block diagram*) za preverjanje skladnosti našega produkta z drugimi sistemi (Futrell et al., 2002, str. 1189–1201; Tepandi, 1997, str. 231–241). S pripravo načrta imamo definirano vse, čemur moramo slediti, da zagotovimo ustrezno kakovost.

S pripravo načrta aktivnosti posegamo tudi v celotni predvideni načrt projekta. To pa pomeni morebitne spremembe pri planu kakovosti, časa in stroškov. Potreben je tudi ponovni pregled

dokumentov zaradi morebitnih sprememb; te so: revizijska poročila kakovosti, plan usposabljanja udeležencev projekta in procesna dokumentacija projekta (Tepandi, 1997, str. 231–241).

Zelo pomemben rezultat procesa zagotavljanja kakovosti so tudi izboljšave, brez katerih v današnjem času nismo konkurenčni. Izboljšava vključuje sprejemanje ukrepov za povečanje učinkovitosti in/ali uspešnosti politik, procesov in postopkov podjetja. Zahteve po spremembah vsebujejo navedbo in specifikacijo posameznih izboljšav. Uporabljajo se v procesu izvajanja integracijskega nadzora kakovosti. Z njihovo pomočjo lahko omogočimo popolno upoštevanje priporočenih izboljšav. Zahteve po spremembah se lahko uporabljajo kot poprava ukrepov ali kot preventivni ukrepi za odpravo napak (PMI, 2008, str. 205).

4 KONTROLA KAKOVOSTI

Pri kontroli kakovosti gre za spremljanje in preverjanje aktivnosti, procesov in rezultatov.

Osredotoča se na spremljanje aktivnosti in rezultata projekta, da zagotovi skladnost projekta s standardi kakovosti. Ko so standardi kakovosti vzpostavljeni, jih moramo spremljati, da zagotovimo doseganje cilja projekta. Kontrola je bistvena za identificiranje problemov, da lahko sprejmemo ukrepe za popravljanje napak in za izboljšave procesov, ki jih nadzorujemo (Marchewka, 2002, str. 241–252). Opravljamo jo skozi celotni življenjski cikel projekta in ga zaključimo šele, ko je naročnik oz. stranka ali projektni sponzor sprejel končne IT rešitve (Kloppenborg & Petrick, 2002, str. 75–85). Vključuje spremljanje in preverjanje oz. kontroliranje aktivnosti povezane s produktom, procesom in projektom (Marchewka, 2002, str. 241–252).

Marchewka (2002, str. 241–252) trdi, da se mora kontrola kakovosti osredotočati na vhode in izhode vsakega procesa. Pri tem poudarja, da če je vhod v proces slabe kakovosti, bo tak tudi izhod, saj proces sam ne more spreminjati inherentne kakovosti izhoda. S tem primerom je želel opozoriti na pomembnost definiranja kakovosti in zahtev že pred začetkom izvajanja posameznega projekta.

Kontrola kakovosti je sistem rutinskih tehničnih aktivnosti, ki merijo in kontrolirajo kakovost produkta med njegovim razvojem. Sistem je zasnovan tako, da zagotavlja rutinsko in dosledno preverjanje celovitosti, pravilnosti in popolnosti podatkov, dokumentiranje in arhiviranje zalog materiala in zapis vseh aktivnosti kontrole kakovosti ter prepoznava in obravnava napake in pomanjkljivosti. Aktivnosti kontrole kakovosti vključujejo splošne metode, kot so: natančno preverjanje pridobivanja podatkov, izračunov in odobreni standardizirani postopki za izračune, meritve, ocenjevanje negotovosti, arhiviranje informacij in poročanje. Višje aktivnosti vključujejo preglede virov kategorij, aktivnosti, metod in dejavnik podatkov (IPCC, 2000).
Del uspešnega managementa kakovosti je tudi kontrola kakovosti, ki jo povezujemo s tistimi sestavnimi deli, ki so uporabljenimi za zagotovitev izpolnjevanja zahtev glede kakovosti. Vključuje tudi vse operativne tehnike in aktivnosti, ki se uporabljajo za izpolnitev podanih zahtev o kakovosti (CIMO guide, 2008, str. 1–18).

Izvajanje kontrole kakovosti je proces opazovanja in zbiranja rezultatov izvedbe aktivnosti za zagotavljanje kakovosti z namenom ocenjevanja uspešnosti in priprave priporočenih sprememb. Nadzor se izvaja ves čas trajanja projekta. Zajema procese projekta in njegove cilje. Kontrolo kakovosti pogosto izvaja oddelek za kakovost ali sorodna enota. Njihova naloga je prepoznavanje slabega izvajanja procesov ali slabe kakovosti produkta in sprejemanje ukrepov za njihovo odpravo. Projektni tim mora imeti za ocenjevanje kakovosti zadostno znanje o statistiki, predvsem o vzorčenju in verjetnosti, da lahko ocenijo rezultat kontrole kakovosti. Koristno je, da dobro poznajo razlike med naslednjimi pari pojmov:

- preprečevanje (izločitev iz procesa) in nadzor (preprečevanje napak stranke),
- vzorčenje z lastnostmi (rezultat je bodisi v skladu ali ni v skladu) in vzorčenje s spremenljivkami (rezultat je ocenjen z vrednostjo na skali, ki meri stopnjo skladnosti),
- toleranca (obseg sprejemljivih rezultatov) in nadzor mejnih vrednosti (mejne vrednosti, ki pokažejo, ali je postopek ušel izpod nadzora).

V nadaljevanju bom predstavila nekaj najpogosteje uporabljenih orodij in tehnik za kontrolo kakovosti.

4.1 Tehnike in orodja kontrole kakovosti

Kontrola kakovosti vključuje številna splošna orodja in tehnike. Schwalbe (2008, str. 299–307) navaja kot najpogosteje uporabljenih sedem Ishikawinih osnovnih orodij kakovosti, statistično vzorčenje in Six Sigma. Opis posameznik tehnik in orodij sledi v nadaljevanju.

4.1.1 Diagram vzrokov in posledic (angl. *Cause and Effect Diagram*)

Diagrame vzroka in posledice imenujemo tudi Ishikawa diagrami ali ribja kost (angl. *fishbone diagram*); ponazarjajo, kako so lahko različni dejavniki povezani v potencialne probleme ali posledice. Možni vzrok lahko odkrijemo z nenehnim spraševanjem »zakaj« ali »kako« po »ribji hrbtenici« (PMI, 2008, str. 208).

Te diagrame se uporablja za raziskovanje vseh potencialnih ali resničnih vzrokov (ali vhodov), ki povzročijo en sam učinek (ali izhod). Vzroki so razvrščeni glede na stopnjo njihove pomembnosti ali podrobnosti, posledica česar je prikazovanje odnosov in hierarhije dogodkov. Tako diagrami vzrokov in posledic pomagajo pri iskanju vzrokov, opredelitvi področij, kjer lahko pride do težav. Omogočajo tudi primerjavo relativnega pomena različnih vzrokov (Basic Tools for Process Improvement – Module 5: Cause and Effect Diagram, 2011; Cause&Effect Diagram, 2011).

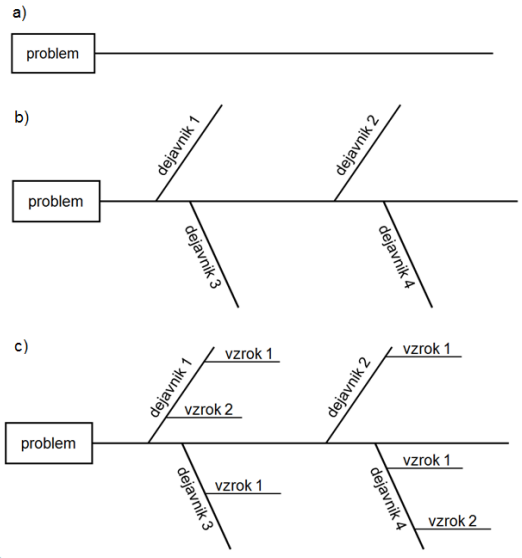
Prednosti izdelave diagrama vzrokov in posledic so (Basic Tools for Process Improvement – Module 5: Cause and Effect Diagram, 2011):

- pomaga določiti osnovne vzroke problema ali lastnosti kakovosti uporabljenega strukturiranega pristopa,
- spodbuja sodelovanje projektnega tima in hkrati izkorišča znanje tima pri izdelavi diagrama,
- uporablja pregledno in urejeno obliko prikaza diagrama,

- prikazuje možne vzroke za spremembe v procesu,
- povečuje poznavanje procesa tako, da pomaga udeležencem projekta izvedeti več o dejavnikih pri delu in kako so le-ti povezani,
- opredeljuje področja, kjer je potrebno zbirati podatke za nadaljnje študije.

Korake nastajanje diagrama prikazuje Slika 3. Prvi korak je identifikacija problema. Prepoznan problem zapišemo v kvadrat na levi strani lista in narišemo »ribjo hrbtenico« (Slika 3 a). Sledi iskanje glavnih dejavnikov za problem – narišemo »kosti« iz »hrbtenice« (Slika 3 b). Tretji korak je iskanje vzrokov po glavnih dejavnikih – črte izhajajo iz »ribjih kosti« (Slika 3 c). Sledi analiza diagrama in iskanje najbolj verjetnega vzroka za problem (Gider, 2011).

Slika 3: Koraki nastajanja diagrama vzrokov in posledic



Vir: Tehnike reševanja problemov, 2011.

4.1.2 Kontrolni diagram (angl. *Control Charts*)

Vsebina je že opisana v magistrskem delu v poglavju 2.1. Gre za statistično orodje, s katerim preučujemo, kako se proces spreminja skozi čas. Podatki so prikazani v časovnem zaporedju. Kontrolni diagram ima vedno sredinsko črto za povprečje, zgornjo črto za zgornjo mejo in spodnjo črto za spodnjo mejo nadzora. Črte se nariše na podlagi podatkov iz preteklosti. S primerjanjem trenutnih podatkov s preteklimi lahko ugotovimo, ali nek proces poteka kot smo predvideli (je pod nadzorom) ali poteka nepredvidljivo (je izpod nadzora) (Data Collection and Analysis Tools, 2011). Cilj uporabe kontrolnega diagrama je doseči stabilen proces, kar pomeni gibanje med zgornjo in spodnjo črto na diagramu. Želimo si doseči sredinsko črto, ki predstavlja povprečje. Takšno gibanje tudi pomeni, da proces poteka pod nadzorom (Basic Tools for Process Improvement – Module 10: Control Chart, 2011; Jarrett, 1995, str. 333–339). Grafično je predstavljeno na Sliki 1 v poglavju 2.1.

4.1.3 Diagram poteka (angl. *Flowcharting, Flowcharts*)

Vsebina je že opisana v poglavju 2.1. Diagram poteka je diagram, ki uporablja grafične simbole za prikaz narave in pretoka korakov posameznega procesa.

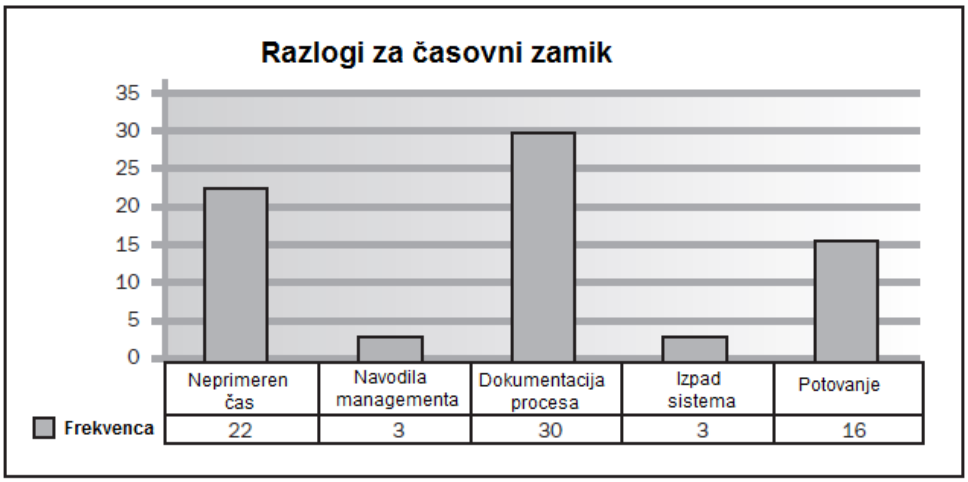
Diagram pomaga razumeti proces in ponuja dodatne poti reševanja korakov. Z njegovo pomočjo ugotovimo, kdo je vključen v proces, katere so teorije o temeljnih vzrokih, načine za poenostavitev procesa, kako izvesti spremembe v procesu, kateri so koraki dodatnih stroškov in zagotavljanje usposabljanja postopka delovanja oziroma kako bi moralo delovati (Basic Tools for Process Improvement – Module 6: Flowchart, 2011; Flow Charts, 2011).

4.1.4 Histogram

Histogram je stolpčni diagram, ki kaže, kako pogosto se je ponovilo stanje spremenljivke oziroma njena porazdelitev. Vsak stolpec predstavlja lastnost problema/rešitve. Višina stolpca predstavlja relativno frekvenco značilnosti (Data Collection and Analysis Tools, 2011; The Histogram, 2011; Schwalbe, 2009, str. 304).

Opisano orodje ponazarja najpogostejši vzrok za težave v postopku s številkami in relativno višino stolpca. Slika 4 prikazuje primer neurejenega histograma, ki prikazuje vzroke zakasnitev projektne skupine (PMI, 2008, str. 210).

Slika 4: Primer histograma – vzroki za zamudo



Vir: PMI, PMBOK, 2008, str. 210.

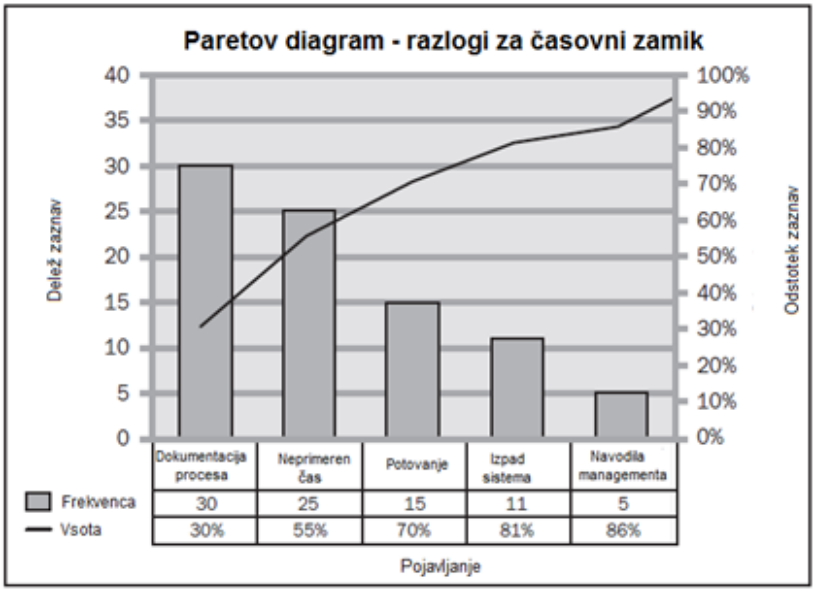
Histogram uporabimo: ko imamo numerične podatke, ko želimo videti obliko porazdelitve podatkov, ko analiziramo, kdaj bo proces dosegel zahteve naročnika, ko želimo distribucijo podatkov pripraviti za razpravo, da je hitra in enostavna itd. (Data Collection and Analysis Tools, 2011).

4.1.5 Pareto diagram (angl. *Pareto Chart*)

Diagram je v 19. stoletju dobil ime po Vifredu Paretu. Pareto je bil ekonomist, ki je domneval, da je velik delež premoženja v lasti majhnega odstotka prebivalstva. To osnovno načelo se kaže v problemih kakovosti (Pareto Chart, 2011). Paretovo načelo pravi, da bo po navadi relativno majhno število vzrokov povzročilo večino težav ali napak. Gre za načelo imenovano 80/20, kjer je 80 % problemov povezanih z 20 % vzrokov (Schwalbe, 2009, str. 304–305).

Paretov diagram je posebna vrsta histograma, ki je urejen po pogostnosti. Kaže, koliko napak je nastalo glede na vrsto ali kategorijo opredeljenega vzroka. Stolpci histograma so urejeni v padajočem vrstnem redu, kar pomeni, da je najvišji stolpec poravnana levo, medtem ko je najnižji poravnan desno (Pareto Chart, 2011). Za osredotočenje na korektivne ukrepe se uporablja rangiranje. Projektna skupina mora najprej obravnavati vzroke, ki povzročajo največje število napak (PMI, 2008, str. 210–211). Za boljšo predstavo sledi grafični prikaz Paretovega diagrama na Sliki 5, ki prikazuje vzroke za zamudo.

Slika 5: Primer Paretovega diagrama – vzroki za zamudo



Vir: PMI, PMBOK, 2008, str. 21.

Opisani diagram največkrat uporabljamo pri analizi o pogostosti ali vzrokih problemov v procesu, ko obstaja veliko problemov in vzrokov ter se želimo osredotočiti na najpomembnejše, ko analiziramo širok spekter vzrokov z iskanjem točno določenih komponent ali za komunikacijo z drugimi o problemih, s katerimi se srečujemo (Pareto, 2011; Pareto Charts, 2011).

4.1.6 Tekoči diagram (angl. *Run Chart*)

Predstavlja najbolj osnovno orodje za prikaz, kako se proces izvaja skozi čas oz. v daljšem časovnem obdobju. Gre za linijski diagram, v katerem so točke, ki predstavljajo podatke, narisane v časovnem zaporedju – to je v zaporedju, kot so se odvijale aktivnosti posameznega procesa. Podatki

predstavljajo točke merjenja, štetja ali prikazovanja izvajanja procesa v odstotkih. Predstavljeni diagram se uporablja predvsem za ocenjevanje in doseganje stabilnosti procesa s poudarkom na posebnih vzrokih sprememb, ki se dogajajo v procesu (Basic Tools for Process Improvement – Module 9: Run Chart, 2011; Siek, 2002).

Tekoči diagram omogoča predstavitev nekaj preprostih statističnih podatkov, ki so povezani s samim procesom (Basic Tools for Process Improvement – Module 9: Run Chart, 2011; Siek, 2002):

- mediana: prikazuje srednjo vrednost predstavljenih podatkov. Uporabimo jo kot središče tekočega diagrama,
- razpon: je razlika med največjo in najmanjšo vrednostjo v podatkih. Potrebujemo in uporabimo ga pri izdelavi ordinatne osi tekočega diagrama.

Prednosti uporabe tekočega diagrama so, da (Basic Tools for Process Improvement – Module 9: Run Chart, 2011):

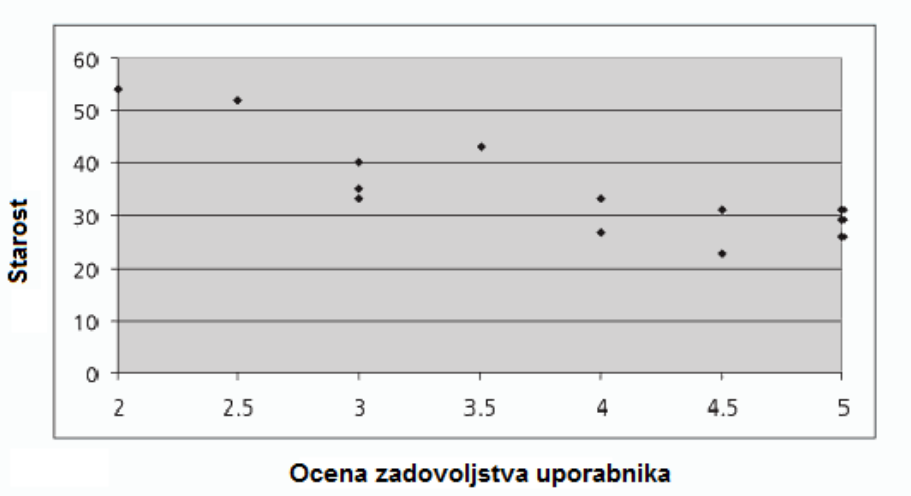
- razumemo razlike v uspešnosti procesa in kako ga lahko še dodatno izboljšamo,
- analiziramo podatke vzorcev, ki niso razvidni iz tabel,
- spremljamo delovanje procesa v daljšem časovnem obdobju, s čimer odkrivamo tudi spremembe,
- imamo možnost razprave, kako proces poteka.

Tekoči diagram je enostavno orodje za uporabo, ki omogoča hitro razumevanje obnašanja posameznega procesa. Uporabljamo ga za preučitev, katere dele procesa se lahko izboljša oz. ali se izboljšave, ki smo jih vključili v proces, obnašajo kot smo predvideli, torej učinkovito (Siek, 2002).

4.1.7 Razsevni diagram (angl. *Scatter Diagram*)

Predstavlja orodje za analizo odnosov med dvema spremenljivkama. Ena predstavlja vertikalno os oz. odvisno spremenljivko, medtem ko druga predstavlja vodoravno os oz. neodvisno spremenljivko (Scatter Diagram, 2011). Bližje kot smo diagonalni črti, bolj sta spremenljivki medsebojno povezani (PMI, 2008, str. 212). Medsebojna povezava ne pomeni nujno neposredne vzročno posledične povezave med spremenljivkama; to drži samo, ko lahko na osnovi vrednosti ene spremenljivke napovemo vrednost druge (Scatter Diagrams, 2011; Schwalbe, 2009, str. 303). Za boljšo predstavo teoretične razlage razsevnega diagram sledi grafični prikaz na Sliki 6.

Slika 6: Primer razsevnega diagrama za odvisnost starosti od zadovoljstva uporabnika



Vir: K. Schwalbe, Information Technology Project Management, 2007, str. 303.

Razsevni diagram največkrat uporabimo, da dokažemo ali ovržemo vzročno posledične odnose (Scatter Diagrams, 2011). Uporabimo ga torej za preučevanje teorije o odnosih med vzroki in posledicami ter za iskanje vzrokov ugotovljenih težav. Poleg tega je uporaba primerna tudi za oblikovanje nadzornega sistema, da bi zagotovili ohranitev koristi izboljšav kakovosti, ki smo jih vključili v izboljšanje kakovosti (Scatter Diagram, 2011).

4.1.8 Inšpekcijski pregled (angl. *Inspection*)

Inšpekcijski pregled je temeljit pregled produkta, da bi ugotovili, ali je le-ta v skladu z dokumentiranimi standardi. Rezultati inšpekcijskih pregledov naj vključujejo predvsem meritve, lete se lahko izvaja na vseh ravneh. Pregleduje se na primer rezultate posamezne aktivnosti ali končni produkt projekta. Inšpekcijskim pregledom lahko pravimo ocene, medsebojni pregledi, revizija ali pregledi. Na nekaterih področjih uporabe ti izrazi predstavljajo zelo ozka in posebna specifična dejstva. Inšpekcijski pregledi se uporabljajo tudi za potrditev odpravljenih napak (PMI, 2008, str. 213).

4.1.9 Ostale tehnike in orodja

Za kontrolo in nadzor kakovosti poznamo še mnogo drugih tehnik in orodij, ki jih nisem podrobneje obravnavala. Nekatere sem že opisala v magistrskem delu, na primer statistično vzorčenje (opisano v poglavju 2.1). Obstajajo še dodatna orodja in tehnike kontrole. Med bolj poznanimi je testiranje, ki ga bom predstavila v nadaljevanju poglavja o kontroli kakovosti.

4.2 Rezultati kontrole kakovosti

Eden izmed glavnih ciljev kontrole kakovosti je doseganje zastavljenih standardov in izboljšanje leteh. Najpomembnejši rezultati tega procesa so odločitve o sprejemljivosti oz. preverjanje produkta, ponovno delo (angl. *rework*) zaradi neustreznosti produkta in meritve aktivnosti procesa (Perform Quality Control, 2011).

V procesu prilagajanja težimo k popravilu in/ali preprečitvi prihodnjih težav kakovosti. Osnova za proces prilagajanja so **meritve** kontrole kakovosti aktivnosti procesa. Vse meritve morajo imeti dosežene rezultate dokumentirane (Schwalbe, 2009, str. 300–306). Z meritvami poskusimo odpraviti težave, ki se pojavljajo v procesu. Tako lahko zagotavljamo kakovost produkta, ki ustreza vsem zahtevam in pričakovanjem naročnika.

Cilj kontrole kakovosti je zagotoviti pravilnost končnega produkta. Ko sprejemamo **odločitve o sprejemljivosti,** imamo dve možnosti: sprejetje ali zavrnitev produkta. Če je produkt sprejet, pomeni, da je naročnik potrdil ustreznost produkta. V nasprotnem primeru, tj. pri zavrnitvi produkta, je potrebno ponovno delo, ker produkt ne ustreza pričakovanjem in zahtevam naročnika (PMI, 2008, str. 213–214). Podjetja težijo k temu, da je potrebnega čim manj ponovnega dela, saj slednje predstavlja velik strošek in porabo časa. Kakovost se s tem namenom najprej planira, nato izvaja, preverja in nadzoruje, da bi končni produkt čim bolj ustrezal zahtevam in pričakovanjem naročnika ter izpolnjeval zahteve povezane s kakovostjo.

Ponovno delo je dejanje, ki je potrebno, ko naročnik zavrne produkt zaradi njegove neustreznosti in neizpolnitve njegovih pričakovanj. Ponovno delo se največkrat pojavlja kot zahtevane spremembe (so potrebne spremembe, da produkt izpolni zahteve) in popravilo odkritih napak (odkrijemo jih s preverjanjem, testiranjem in izvajanjem preventivnih akcij za doseganje zahtevne kvalitete) (Schwalbe, 2009, str. 300). Ponovno delo je navadno zelo drago, kar predstavlja bistveno povečanje stroškov projekta glede na načrtovane.

Poleg že navedenih rezultatov kontrole kakovosti je rezultat tudi posodobitev določenih dokumentov in že izdelanih načrtov projekta izvajanja kakovosti (PMI, 2008, str. 213).

4.3 Six Sigma

Začetki Six Sigme segajo v začetek in sredino osemdesetih let. Razvila se je v podjetju Motorola zaradi zelo slabe kakovosti njihovih izdelkov (Poslovna strategija SIX SIGMA kaj je to, 2011). Kasneje je pri razvoju načel Six Sigme sodelovalo veliko strokovnjakov s področja managementa. Gre za celovito strategijo oz. filozofijo podjetja, v ožjem smislu pa jo lahko obravnavamo kot metodo managementa projektov za izboljšanje kakovosti produktov. Pri tej strategiji gre za koncept, ki je usmerjen h kupcem in je podkrepljen s podatki in statističnimi metodami (Uspešna vpeljava, 2011; Schwalbe, 2009, str. 307–312).

Six Sigma je osredotočena in zelo učinkovita vpeljava principov in orodij za kakovost. Njena vizija je usmerjanje k procesom brez napak. Sigma je dejansko grška črka, ki se jo v statistiki uporablja za merjenje sprememb. V poslovnem svetu se uporablja za merjenje sprememb v procesih. S pridobljenimi rezultati skušajo podjetja svoje poslovne procese čim bolj stabilizirati (Poslovna strategija SIX SIGMA kaj je to, 2011; Schwalbe, 2009, str. 307–312).

Znotraj metodologije Six Sigma obstaja več sistematik za reševanje problemov in izboljšanje procesov. Širše priznani in standardizirani sta DMAIC (angl. *Define-Measure-Analyze-Improve-*

Control) in DFSS (angl. *Design For Six Sigma*). Poleg teh obstaja vrsta različic, ki so si jih izmislila podjetja za reševanje specifičnih težav, s katerimi so se srečevala (Six Sigma Traning Resources, 2011; Uspešna vpeljava, 2011).

V nadaljevanju bom podrobneje obravnavala izboljšavo procesov podjetja imenovano DMAIC. V prvem koraku, v fazi **definiranja** (angl. *Define*), se določi, v katerih mejah se bo obravnaval problem, analiziral potencial in finančni prihranek, plan izvedbe projekta in se določi tim za delo na projektu. Izdela se tudi prvi grobi vpogled v raziskovani proces z identifikacijo vhodov, izhodov in njihovih merljivih karakteristik, katerih vpliv dokažemo ali ovržemo v fazi analiziranja (Six Sigma Traning Resources, 2011; Uspešna vpeljava, 2011).

V fazi **merjenja** (angl. *Measure*) se definirajo metrike. Temu sledijo zbiranje, priprava in prikaz podatkov na primeren način. Brez dobro načrtovanega in izvedenega tega koraka je kasneje v koraku analize skoraj nemogoče ali zelo težko poiskati enačbo soodvisnosti med najpomembnejšimi vhodi in izhodi, kar je glavni cilj Six Sigma projekta. V ta namen se najprej določi možne vzroke težav in postavi hipoteze o posameznih in povezanih vplivih. Sledi določitev merilnih mest v procesu, velikosti vzorcev za merjenje in preverjanje sposobnosti merilnega sistema. Sledijo meritve in zbiranje podatkov. Zaključek faze predstavljajo prve izračunane karakteristike (kot je npr. stopnja sigme) ključnih izhodov procesa (Six Sigma Traning Resources, 2011; Uspešna vpeljava, 2011).

Na podlagi zbranih podatkov se s pomočjo statističnih metod preveri hipoteze iz prvega koraka in zmanjša število ključnih vplivnih vhodov na izhod. Tu je bistveno, da se poišče in dokaže, katere vhodne spremenljivke imajo največji vpliv na izhod. Nato raziščemo soodvisnost med vhodi in izhodi, kar privede do optimalnih nastavitev za želeni izhod procesa. V fazi **analiziranja** (angl. *Analyse*) in v predhodni fazi se v večini primerov konvencionalnega reševanja problemov posveti premalo pozornosti, kar vodi k neoptimalnim rešitvam, ki so pogosto rezultat ugibanja in poskušanja (Six Sigma Traning Resources, 2011; Uspešna vpeljava, 2011; Schwalbe, 2009, str. 307–312).

Predzadnja faza je faza **izboljšanja** (angl. *Improve*). V tej fazi je najpomembnejša kreativnost članov tima in poznavalcev procesa. Uporabljajo se lahko vse bolj ali manj znane kreativne tehnike iskanja izboljšav (viharjenje možganov, 6-3-5 metoda, De Bono, morfološka sestavljanka, postavitev na glavo, scamper itd.). Po analizi in ovrednotenju rešitev se izdela plan vpeljave najprimernejše rešitve. Te se v tem koraku še ne vpelje v proces, kljub imenu, ki ga ta faza nosi (Six Sigma Traning Resources, 2011; Uspešna vpeljava, 2011).

Zadnja faza je **nadzorovanje** (angl. *Control*), kjer se nadaljuje meritve, s katerimi smo začeli v fazi meritev. To je tudi faza vpeljave rešitev, ki smo jih našli v predhodni fazi. S primerjavo meritev v obeh fazah dokažemo razliko po izvedbi sprememb. Sledi iskanje možnosti prenosa rešitve tudi na ostale procese v podjetju in predaja dokumentacije lastniku procesa. Po uspešno zaključenem projektu se navadno s finančnimi in nefinančnimi vzpodbudami tim nagradi za uspešno in v skladu

z začetnimi zahtevami izveden projekt ter vložen trud in napor v času trajanja projekta (Six Sigma Traning Resources, 2011; Uspešna vpeljava, 2011).

Six Sigma je poslovna filozofija, ki je osredotočena na kupca in si prizadeva zmanjšati potratnost, povišati raven kakovosti in izboljšati finančno uspešnost. Podjetje, ki uporablja to poslovno strategijo, si zastavlja visoke cilje in uporablja že opisano metodo DMAIC za dosego nadpovprečnega izboljšanja kakovosti. Sponzor projekta je v konceptu Six Sigma poimenovan kot prvak (angl. *champion*). Posebno ime imajo tudi projektni managerji, to so managerji timov (Schwalbe, 2009, str. 307–312).

Izkušnje podjetij, ki uporabljajo strategijo Six Sigma, so zelo pozitivne. Večina jih opazi, kako veliko prihranijo pri stroških. Najbolj očitno se pozna prihranek pri stroških slabe kakovosti, ki so za delo lahko visoki. Pri večini podjetij to predstavlja 25 do 40 % prihodkov, medtem ko je ta številka pri projektih, ki uporabljajo Six Sigma manj kot 5 %. Strokovnjaki opozarjajo, da do tega pride zaradi dviga stopnje sigme, ki posledično občutno zniža število napak. Dodajajo še, da se pozna uporaba obravnavane strategije, ko napake preučujemo v milijonih in ne v tisočih. Poznavalci tako podjetjem priporočajo, da pred uvedbo načel Six Sigma nekdo dejansko razume vsa navedena načela. Najpogostejša napaka hitrega uvajanja je merjenje napačnih procesov. Nastanejo nepotrebne meritve, ki povzročajo izgubo časa in zmedenost pri strateških ciljih (Uspešna vpeljava, 2011; Schwalbe, 2009, str. 307–312).

4.4 Testiranje

Mnogi strokovnjaki s področja informacijske tehnologije so mnenja, da testiranje pride na vrsto šele ob koncu procesa razvoja produkta (Schwalbe, 2009, str. 313–315). Po daljšem brskanju literature in prebiranju člankov na temo testiranja, sem ugotovila, da temu ni tako. Če želimo zagotoviti kakovostno testiranje in z njim tudi doseči svoj namen, je testiranje potrebno obravnavati že bistveno prej, saj igra pomembno vlogo pri doseganju in ocenjevanju kakovosti programske opreme – produkta.

Naik in Tripathy (2008, str. 7, 84) se strinjata z avtorjema, kot sta Friedman in Voas (Naik & Tripathy, 2008, str. 7, 84), da je testiranje postopek preverjanja kakovosti programske opreme – produkta. S pomočjo preverjanja lahko produkt ocenimo in zanj poiščemo tudi izboljšave.

Za uspešno testiranje produkta na področju programske opreme je potrebno, da smo s planiranjem začeli pravi čas, upoštevali vse zahteve in specifikacije naročnika, predvideno smer delovanja produkta, upoštevali programski jezik uporabljen za razvoj programske opreme itd. Več o tem sledi v nadaljevanju.

4.4.1 Osnovni koncept teorije testiranja

Ideja testiranja programske opreme (angl. *software*) je stara ravno toliko, kot je staro računalniško programiranje. Začetki segajo v začetek 60 let prejšnjega stoletja, ko je programska oprema postajala vse večja in kompleksnejša. Prav zato se je pojavila potreba po sistematičnem

odstranjevanju napak na tem področju. Tako so raziskovalne skupnosti in programerji začeli posvečati več pozornosti testiranju programske opreme. V ta namen so v zgodnjih 70 letih prejšnjega stoletja odprli novo poglavje na tem področju, ki so ga poimenovali Teorija testiranja (Naik & Tripathy, 2008, str. 14). Ta teorija daje poudarek (Naik & Tripathy, 2008, str. 14–16):

- odkrivanju napak preko testiranja, ki temelji na izvajanju,
- oblikovanju testnih primerov iz različnih virov, kot so: zahteve definirane v specifikaciji, izvorna koda ter vhodi in izhodi programov,
- oblikovanju skupine testnih primerov iz nabora vseh možnih testnih primerov,
- učinkovitosti izbrane strategije testnih primerov,
- uporabi testnih načrtovalcev med testiranjem,
- prednosti izvajanja izbrane skupine testnih primerov,
- ustreznosti analize testnih primerov.

Teoretični temelj za testiranje daje testerjem in razvijalcem dragocen vpogled v programsko opremo, sisteme in razvoj procesov, kar je privedlo do oblikovanja učinkovitejših testnih primerov po nižji ceni. Ob upoštevanju testiranja lahko teoretično pride do povečanega pričakovanja pri odkrivanju napak. Izoblikuje se želja, da je v posamezni programski opremi mogoče najti vse napake, kar pa predvideva le teorija, saj tega zaradi prevelike kompleksnosti in obsežnosti programov v realnosti ni mogoče. Vsaka teorija se srečuje tudi z omejitvami, tako so omejitve pri teoriji testiranja najbolje opisane s strani Dijkstra, ki pravi: »Testiranje lahko odkriva prisotnost napak, ne odkriva pa njihove odsotnosti!« (Naik & Tripathy, 2008, str. 22–24).

4.4.2 Celovito testiranje

Celovito testiranje pomeni, da na koncu faze testiranja v programu ni neodkritih napak. Vsi problemi, s katerimi bi se programska oprema lahko srečevala, morajo biti v tej fazi znani. Pri večini programov je doseganje celovitega testiranja skoraj ali celo nemogoče. Razlogi za to so preobsežen nabor vseh možnih vhodov, prekompleksna struktura programa in nezmožnost predvideti vse zunanje dejavnike, ki vplivajo na uporabo razvite programske opreme (Naik & Tripathy, 2008, str. 13).

4.4.3 Aktivnosti testiranja

Za preizkus programske opreme mora testni inženir izvesti zaporedje testnih aktivnosti. Te so podrobneje obrazložene v nadaljevanju. Razlage se osredotočajo samo na en testni primer.

Prva aktivnost je **določitev cilja**, ki ga želimo testirati. Cilj opredeljuje namero in namen za oblikovanje enega ali več testnih primerov, da se prepričamo v delovanje programa in njegovo dosego cilja. Z vsakim testnim primerom mora biti povezan jasen namen testiranja (Pezzè & Young, 2007; Ammann & Offutt, 2008, str. 3–11).

Druga aktivnost je **izbira vhodov** v program. Ta navadno temelji na zahtevah, ki so navedene v specifikaciji programske opreme, izvorni kodi ali pričakovanjih. Pri izbiri vhodov moramo

upoštevati cilj testa (Naik & Tripathy, 2008, str. 14–17; Pezzè & Young, 2007; Ammann & Offutt, 2008, str. 3–11).

Sledi **izračun pričakovanega rezultata** programske opreme z že izbranimi vhodi. V večini primerov to lahko storimo šele takrat, ko zelo dobro poznamo in razumemo cilj testa in specifikacijo programske opreme, ki je vključena v test (Naik & Tripathy, 2008, str. 14–17; Pezzè & Young, 2007; Ammann & Offutt, 2008, str. 3–11).

Četrta aktivnost zajema **pripravo pravega okolja za delovanje** programske opreme. V tem koraku morajo biti izpolnjene vse zunanje predpostavke programa, kot sta na primer vzpostavitev lokalnega (vzpostavitev omrežne povezave, dostop do prave baze podatkov itd.) in oddaljenega sistema (vzpostavitev povezave s strežnikom, če testiramo programsko opremo na strani odjemalca), ki nista del programske opreme (Naik & Tripathy, 2008, str. 14–17; Pezzè & Young, 2007; Ammann & Offutt, 2008, str. 3–11).

Pri peti aktivnosti preverjamo **izvajanje programske opreme**. Testni inženir preveri delovanje programa z izbranimi vhodi in opazuje dejanske rezultate, ki jih program izvede. Za izvedbo testnega primera morajo biti zagotovljeni vhodi na različnih fizičnih lokacijah in ob različnem času. Ta koncept je namenjen predvsem usklajevanju različnih komponent testnega primera (Naik & Tripathy, 2008, str. 14–17; Pezzè & Young, 2007; Ammann & Offutt, 2008, str. 3–11).

Zadnja aktivnost je **analiza testnih podatkov.** Glavna naloga analize je primerjava dejanskega izida s pričakovanim izidom delovanja programske opreme. Kompleksnost primerjave je odvisna od kompleksnosti vhodnih podatkov. Ob zaključku te aktivnosti podamo oceno izvedenega testa. Poznamo tri vrste ocen: uspešno (angl. *pass*), neuspešno (angl. *fail*) in nedefinirano (angl. *inconclusive*). Če je program ocenjen z uspešno, pomeni, da je izid tak, kot smo pričakovali. Nasprotno velja za neuspešno oceno. Nedefinirano pa pomeni, da so potrebni dodatni testi za opredelitev uspešnosti oz. neuspešnosti (Naik & Tripathy, 2008, str. 14–17; Pezzè & Young, 2007; Ammann & Offutt, 2008, str. 3–11).

4.4.4 Testni primer (angl. *Test Case*)

IEEE Standard 610 (1991) definira testni primer kot: »(1) nabor testnih vhodov, pogojev za izvajanje in pričakovanih rezultatov, ki so razviti za točno določen cilj, kot je izvajanje določene programske poti (angl. *program path*) ali preverjanje skladnosti s točno določeno zahtevo; (2) Gre za dokumentacijo določenih vhodov, predvidenih rezultatov in nabor pogojev izvajanja za testni primer« (Kerner, 2003). Po mnenju Kernerja (2003) je testni primer vprašanje, ki ga zastavimo programski opremi – programu. Glavni namen izvedbe testa je pridobiti informacije, na primer ali bo program pri izvajanju testa uspešen ali neuspešen.

Patton (2005, str. 277–280) definira testni primer takole: »Testni primeri so specifični vhodi, ki jih bomo preizkusili, in postopki, ki jim bomo sledili, ko bomo testirali programsko opremo.«

Testni primer je v najosnovnejši obliki preprost par vhoda in pričakovanega izida. Ločimo dva sistema, v katerih se izvaja teste – in to sta sistem brez stanja (angl. *stateless*) in sistem s stanjem (angl. *state-oriented*). V **sistemu brez stanja** je izid odvisen samo od trenutnega vhoda. Testni primeri za tak sistem imajo zelo enostavno strukturo. V **sistemu s stanjem** je izid odvisen od trenutnega vhoda in od trenutnega stanja sistema. Testni primer za ta sistem vsebuje oblike odločitev in čas za zagotavljanje vhoda v sistem (Naik & Tripathy, 2008, str. 11).

4.4.4.1 Potrebne informacij za izbiro testnega primera

Oblikovanje testnih primerov bo še naprej ostalo v središču raziskovalnih skupnosti in programerjev. Proces razvoja programske opreme ustvarja obsežne informacije, kot so določitev zahtev, oblikovanje dokumenta ali izvorna koda. Za načrtovanje učinkovitih testov po nizki ceni testni načrtovalci analizirajo naslednje informacije: zahteve in specifikacije funkcionalnosti, izvorno kodo, vhod in izhod delovnih področij, operativni profil in model napak (Naik & Tripathy, 2008, str. 18–21; Ammann & Offutt, 2008, str. 3–11).

- **Zahteve in specifikacije funkcionalnosti**
  Proces razvoja programske opreme se začne z zajemanjem potreb uporabnikov. Narava in količina potreb se določijo na začetku razvoja sistema, ki bo odvisen od točno določenega modela življenjskega cikla, kateremu bo sledil. Tako recimo v primeru modela Waterfall poskusi inženir za razvoj programske opreme zajeti v test večino podanih zahtev, medtem ko na primer v modelu Scrum testni inženir zajame na začetku v test samo nekaj zahtev (Naik & Tripathy, 2008, str. 18–21). Testni inženir zato zajame vse zahteve, ki jih mora programska oprema izpolnjevati ne glede na izbrani model življenjskega cikla.

  Zahteve so lahko določene na **neformalen** način, to je lahko kombinacija golega besedila (angl. *plaintext*), enačb, grafičnih prikazov in diagramov poteka. Čeprav je neformalen način predstavitve zahtev lahko dvoumen, je naročniku enostavno razumljiv. Včasih so zahteve določene na **formalen** način. Tak primer sta Estelle (formalna opisna tehnika) in avtomat s končnim številom stanj (angl. *finite-state machine*). Formalni in neformalni način določitve zahtev programske opreme sta za testni primer primarna vira informacij (Naik & Tripathy, 2008, str. 18–21; Mathur, 2008, str. 106–108).

- **Izvorna koda (angl. *Source Code*)**
  Izvorna koda je skupina navodil, ki je napisana v izbranem programskem jeziku; ta se kasneje prevede v strojni jezik. Le-ta teče na računalniku v obliki programske opreme. Specifikacija zahtev opisuje pričakovano obnašanje sistema, izvorna koda pa opisuje dejansko obnašanje sistema. Ne glede na podrobnost načrta, ki ga pripravi načrtovalec programa, lahko programer v sistem skozi izvorno kodo vključi dodatne podrobnosti, zato mora biti testni primer zasnovan na osnovi izvorne kode (Naik & Tripathy, 2008, str. 19; Ammann & Offutt, 2008, str. 3–11).

- **Vhod in izhod delovnih področij (angl. *Input and Output Domains*)**
  Nekatere vhodne vrednosti programske opreme imajo poseben pomen, zato jih je potrebno obravnavati ločeno. Včasih imajo tudi izhodne vrednosti poseben pomen; torej je potrebno programsko opremo testirati, da preverimo, ali le-ta ustvari posebne vrednosti za vse možne vzroke. Pri oblikovanju testnega primera in testiranja programske opreme je potrebno upoštevati vse posebne vrednosti vhodov in izhodov delovnega področja programske opreme (Naik & Tripathy, 2008, str. 19).

- **Operativni profil**
  Gre za kvantitativno opredelitev, kako bo sistem uporabljen. Ustvarjen je bil za usmerjanje testnih inženirjev pri izbiri testnih primerov z uporabo vzorcev rabe sistema. Operativni profil je razvil Mills na IBM v okviru »Čistilne sobe programske opreme (angl. *Cleanroom Software Engineering*)« in Musa pri AT & T Bell Laboratories za pomoč pri razvoju boljše zanesljivosti programske opreme (Naik & Tripathy, 2008, str. 19–20). Ideja operativnega profila je izpeljati bodočo zanesljivost programske opreme s pomočjo opazovanih testnih rezultatov, ko je program že v uporabi. Za natančno oceno zanesljivosti sistema je pomembno, da pri testiranju upoštevamo predvsem, kako se bo programska oprema dejansko uporabljala. Ta koncept se uporablja zlasti pri testiranju spletnih aplikacij, kjer se podatki uporabnikov zberejo s spletnih strežnikov. Z zbranimi podatki se oblikuje testne primere (Sampath, Sprenkle, Gibson, Pollock, & Greenwald, 2007, str. 643–657).

- **Model napak (angl. *Fault Model*)**
  Odličen vir informacij za oblikovanje novih testnih primerov so pojavljajoče napake. Razvrstimo jih lahko v tri različne razrede: vzpostavitvene napake, logične napake in napake vmesnikov. Vse se shranjujejo v repozitoriju, ki je neke vrste odlagališče podatkov. Testni inženirji lahko s pomočjo teh podatkov oblikujejo teste, s katerimi zagotovijo odpravo omenjenih vrst napak v programski opremi (Naik & Tripathy, 2008, str. 20).

  Na podlagi teh napak ločimo tri tipe testiranja: ugibanje napak, vstavljanje napak in analiza mutacij. Pri **ugibanju napak** oblikovalec testnih primerov uporabi svoje izkušnje, da oceni trenutno stanje, in ugiba, kje in kakšne vrste napak se pojavljajo; za tem oblikuje teste, s katerimi te napake izpostavi. Pri **vstavljanju napak** se znane napake vstavi v programsko opremo in požene testno orodje. S tem preverimo učinkovitost testnega orodja, ki bi moralo vstavljene napake zaznati. Če orodje te napake najde, predvidevamo, da bo našel tudi druge, ki se pojavljajo v programski opremi. **Analizo mutacij** imenujemo tudi simulacija napak. Gre za to, da izbrane dele programa delno spremenimo in nato poženemo test z namenom določanja zmogljivosti odkrivanja napak testnega orodja. Ni nujno, da rezultat testa zazna narejene spremembe v programski opremi, saj smo s spremembo lahko program celo izboljšali (Naik & Tripathy, 2008, str. 20).

4.4.5 Testiranje na osnovi modela

Gre za tehniko testiranja, ki je namenjena avtomatskemu oblikovanju testov s pomočjo modelov; oblikujemo jih iz elementov, ki so pridobljeni v celotnem procesu razvoja programske opreme (Dalal et al., 1999, str. 1–10). Ne moremo testirati brez razumevanja, kaj naj bi programska oprema počela. Kompleksnost le-te predstavimo s pomočjo modelov, s katerimi si pomagamo oblikovati teste. Ta vrsta tehnike predlaga uvedbo prednosti procesa testiranja za samo programsko opremo in organizacijo. Prednosti so: krajši razporedi testiranja, nižji stroški, boljša kakovost, povečana komunikacija med razvijalci in testerji, zgodnje izpostavljanje nejasnosti tako v specifikaciji kot pri oblikovanju, omogočanje samodejnega ustvarjanja neponavljajočih in koristnih testov, možnost samodejnega poganjanja testov ter enostavno posodabljanje testnih orodij za spremenjene zahteve (Dias-Neto & Travassos, 2009, str. 1487–1504).

Dias-Neto in Travassos (2009, str. 1487–1504) menita, da je glavni dejavnik, ki vpliva na testiranje, skupno število testnih primerov. Za vsak testni primer je potrebno določiti razporejanje virov za načrtovanje, oblikovanje in izvedbo. Samodejno oblikovanje testnih primerov je zanimivo zato, ker niža stroške in priprave na testiranje. Model testiranja zajema sistematično (preizkušena je vsaka ciljna kombinacija), osredotočeno (iščemo, kje bi se lahko najprej pojavili hrošči/napake) in samodejno (izdelava in izvajanje velikega števila doslednih in ponavljajočih testov) testiranje.

Pristopi za testiranje na osnovi modela se navadno ukvarjajo z različnimi nivoji abstrakcije, vedenjskimi/strukturnimi modeli (uporabljajo se za oblikovanje testov), odnosi med modeli in izvorno kodo, tehnikami za oblikovanje testnih primerov, merili za izbor testa in razpravo o tem, kateri del testiranja naj bo avtomatiziran. Pristope lahko med seboj ločimo s pomočjo poenotenega jezika za modeliranje (angl. *Unified Modelling Language – UML*), ki je postal standard za objektno usmerjeno modeliranje (Dias-Neto & Travassos, 2009, str. 1487–1504). Glavni namen poenotenega jezika za modeliranje – UML je zagotavljanje jezika z enotnim konceptom in stabilnostjo, ki ga lahko razvijalci uporabijo za razvoj in izgradnjo računalniških aplikacij (Bell, 2003).

Strokovnjaki poudarjajo, da je uspeh testiranja neposredno povezan s kakovostjo elementov, ki so pri oblikovanju vedenjskih/strukturnih modelov za testiranje uporabljeni v obliki vhoda. Če modele oblikujemo iz nepopolnih, neusklajenih ali napačnih elementov, je rezultat izvajanje neprimernih testov in vrednotenje napačnih značilnosti programske opreme. Zato je potrebno elemente pred uporabo pregledati. S tem lahko povečamo možnost zagotavljanja zastavljene ravni kakovosti samodejnega testiranja (Dias-Neto & Travassos, 2009, str. 1487–1504; Baresi & Pezzè, 2006).

Potrebno se je zavedati, da se vsega ne da testirati samodejno. Zato je v testiranje potrebno vključiti ročno izvajanje testnih primerov, s katerimi navadno želimo oceniti posebne značilnosti ali lastnosti programske opreme, ki niso zajete v samodejne testne primere. Posledica vpeljave te oblike testiranja je povečanje kakovosti rezultatov, ki jih dobimo po izvedbi obeh testiranj (Dalal et al., 1999, str. 1–10).

4.4.6 Ravni testiranja

Testiranje se izvaja na različnih ravneh, ki vključujejo bodisi celoten sistem bodisi samo dele sistema skozi celoten življenjski cikel programske opreme. Preden je sistem programske opreme dejansko uporabljen, gre skozi štiri faze testiranja. Te so testiranje enot, integracije, sistema in sprejemljivosti. Prve tri ravni se odvijajo v okviru podjetja, ki razvija programsko opremo. V njihovo izvajanje so vključeni številni udeleženci, kot so programerji, testerji, projektni vodja itd. Testiranje sprejemljivosti opravlja naročnik programske opreme in se ga opravi nazadnje (Naik & Tripathy, 2008, str. 16–18; Baresi & Pezzè, 2006; Schwalbe, 2009, str. 313–315).

- **Test enot (angl. *Unit Test*)**
  To je test enot ali modulov, kot jih poimenujeta avtorja Baresi in Pezzè (2006). V prvi fazi programerji testirajo posamezne enote oz. module programa v izolaciji. Enote oz. moduli so postopki, funkcije, metode ali razredi (Naik & Tripathy, 2008, str. 16, 51–88). V tej fazi je naloga programerja zagotavljanje, da posamezna enota programa deluje v zadovoljivem obsegu. V fazi testiranja enot je zelo pomembno, da ugotovimo in odstranimo napake, ki so se pojavile, saj bi jih bilo v kasnejših fazah razvoja težje najti in dražje odpraviti (Baresi & Pezzè, 2006). Ko posamezne enote uspešno opravijo test enot, so pripravljene za sestavo v večje podsisteme, imenovane integracije (Naik & Tripathy, 2008, str. 16).

- **Test integracije (angl. *Integration Test*)**
  Pojavi se med testom enot in testom sistema, da preveri funkcionalnost enot združenih v podsisteme. Veliko strategij testiranja integracij predlaga, da testiramo integrirane enote oz. module s postopnim večanjem (angl. *incrementally*) (Baresi & Pezzè, 2006). Test integracije zagotavlja, da posamezen podsistem deluje pričakovano in ima odstranjene vse napake, ki so se pojavile v času testiranja. Po uspešno opravljenem testu integracije so podsistemi pripravljeni za združitev v skupni sistem (Schwalbe, 2009, str. 313–315; Naik & Tripathy, 2008, str. 16–18, 158–192).

- **Test sistema (angl. *System Test*)**
  Testiranje sistema je kritična faza procesa razvoja programske opreme, kajti bliža se rok dokončanja, obenem pa si programer v tem času želi najti čim več napak in jih odpraviti (Naik & Tripathy, 2008, str. 16–18, 408–450). Testiranje sistema preveri skladnost med celotnim sistemom in specifikacijami sistema (Baresi & Pezzè, 2006). Opisano testiranje zajema številne različne aktivnosti, kot so: priprava plana testiranja, oblikovanje testnega orodja, priprava testnega okolja, izvajanje testov s sledenjem jasni strategiji in spremljanje procesa testiranja (Naik & Tripathy, 2008, str. 16–18, 408–450).

- **Testiranje regresije (angl. *Regression testing*)**
  Je raven testiranja, ki se izvaja skozi celoten življenjski cikel razvoja programske opreme. Testiranje regresije je prisotno vsakič, ko spremenimo ali nadgradimo kakršnokoli enoto oz. modul programske opreme. Ključna ideja tega testiranja je preverjanje uvedenih sprememb in njihovo vplivanje na obstoječe enote oz. module. Preveriti želimo, ali je uvedba spremembe

povzročila napake na obstoječih (starih) delih programa, ki niso bili spremenjeni. Testiranje regresije ne predstavlja ločene ravni testiranja, temveč je zgoraj testiranje opisanih faz (test enot, integracije in sistema) (Naik & Tripathy, 2008, str. 17, 194, 214–215; Baresi & Pezzè, 2006).

Pri tem testiranju ne oblikujemo novih testnih primerov. Uporabimo tiste, ki jih izberemo iz obstoječega nabora, da preverimo delovanje nove različice programske opreme. Ker je to testiranje drago in vzame veliko časa, se poskuša znižati stroške in prihraniti čas z izvedbo samo izbranih testnih primerov. Pri tem se pojavlja vprašanje, katere testne primere in koliko jih izbrati iz obstoječega nabora, da bodo le-ti najbolj verjetno prikazali morebitne napake nove različice (Naik & Tripathy, 2008, str. 17, 194, 404–405). Avtorja Baresi in Pezzè (2006) pa sta mnenja, da je to faza ponovnega testiranja (angl. *re-test all*). Vanjo morajo po njunem mnenju biti vključeni vsi do sedaj opravljeni testi na programski opremi in ne le izbrani.

- **Test sprejemljivosti (angl. *Acceptance Test*)**
  Po zaključku testiranj na ravni sistema je programska oprema pripravljena za naročnika. Ta nato opravi vrsto testov, ki jih poznamo pod imenom testiranje sprejemljivosti ali test sprejemljivosti. Cilj testiranja je merjenje kakovosti programske opreme in ne iskanje napak, kar je cilj testiranja sistema. Zelo pomemben del tega testiranja je tudi zadovoljitev pričakovanj naročnika, ki jih je določil pred začetkom razvoja programske opreme (Naik & Tripathy, 2008, str. 17–18, 450–471).

  Ricca et al. (2009, str. 270–283) definirajo test sprejemljivosti, kar je po definiciji IEEE Standarda 1012 iz leta 1986, ki pravi, da je to formalno testiranje, ki se izvaja za ugotovitev, ali sistem izpolnjuje njegove kriterije sprejemljivosti in omogoča naročniku sprejetje ali zavrnitev razvite programske opreme. Testiranje se navadno izvaja na celotnem sistemu ali večinskem delu sistema. Pripravo testa sprejemljivosti si lahko olajšamo z različnimi temu namenjenimi orodji. Eno takih je tudi Fit tabela, ki predstavlja okvir za test integriranosti (angl. *Framework for Inegrated Test*). Omenjeni avtorji poudarjajo, da taka orodja olajšajo izdelavo testa sprejemljivosti ter tudi razjasnijo določene zahteve, ki jih naročnik postavi zaradi svoje enostavnosti uporabe in razumevanja.

  Avtorja Naik in Tripathy (2008, str. 17–18, 450–471) ločita dve vrsti testa sprejemljivosti, in sicer: uporabniški test sprejemljivosti (angl. *User acceptance test*) in poslovni test sprejemljivosti (angl. *Business acceptance test*). **Uporabniški test sprejemljivosti** se izvaja s strani naročnika z namenom zagotavljanja ustreznosti programske opreme. Ustreznost pomeni izpolnjevanje pogodbenih kriterijev sprejemljivosti za prevzem produkta. Kriteriji sprejemljivosti (angl. *Acceptance criteria*) vključujejo funkcionalno pravilnost in dovršenost, natančnost, podatkovno celovitost, pretvorbo podatkov, varnostne kopije in obnovitev podatkov, konkurenčnost, uporabnost, zmogljivost, potrebni čas za zagon, nivo obremenitve, zanesljivost in razpoložljivost, možnost vzdrževanja in servisiranja, robustnost, odzivnost itd. **Poslovni test sprejemljivosti** se izvaja v okviru podjetja, ki razvija programsko opremo. Namen testa je

zagotoviti, da bo programska oprema dejansko uspešno opravila uporabniški test sprejemljivosti.

4.4.7 Planiranje testiranja

Testiranje se začne s planiranjem, s katerim si zagotovimo organizacijo izvedbe posameznih aktivnosti testiranja. Načrt nudi okvir in obseg testiranja, podrobnosti potrebovanih virov, potreben čas, urnik izvajanja in stroške. V načrt vključimo uvod (zajema ime projektnega testa, zgodovino sprememb, terminologijo in definicije, reference, imena in datume odobritve načrta in povzetek preostalega plana testiranja), opis funkcionalnosti (povzema funkcije sistema, ki jih bo testiranje zajemalo), predpostavke (določimo predpostavke, zaradi katerih testiranje določenih funkcionalnosti ne bo mogoče), pristop k testiranju (definiramo, kako se bomo testiranja lotili), strukturo testnih primerov (definira strukturo in cilje testnih primerov, ki so vključeni v testiranje), testno okolje (zagotovitev ustreznega testnega okolja, ki naj bo čim bolj podobno tistemu, kjer se bo programska oprema uporabljala), strategijo izvajanja testiranja (predstavlja izbiro strategije, po kateri bo testiranje potekalo), oceno zahtevnosti testa (podaja oceno količine potrebnega dela za izvedbo testiranja), urnike in mejnike testiranja (prikazujejo potek testiranja in datume, do katerih mora biti testiranje opravljeno) (Naik & Tripathy, 2008, str. 21–22, 355–408).

4.4.8 Testiranje bele in črne skrinjice

Testne primere oblikujemo na podlagi različnih informacij, ki prihajajo iz virov: specifikacije, izvorne kode in posebne lastnosti vhodov in izhodov programske opreme. Vsi ti viri nudijo testnim inženirjem dopolnilne informacije za oblikovanje testnih primerov. Na podlagi tega poznamo dva širša koncepta testiranja, to sta test bele (angl. *white-box testing*) in črne skrinjice (angl. *black-box testing*) (Naik & Tripathy, 2008, str. 20–21, 42; Kobayashi, Tsuchiya, & Kikuno, 2002, str. 113–121; Cai, Li, & Ning, 2005, str. 552–579).

Testiranje bele skrinjice označujemo tudi kot strukturirano testiranje, ki uporablja interne detajle programske opreme (kot so izvorna koda, potek kontrole in tok podatkov) za oblikovanje testnih primerov. Najpogosteje se to testiranje izvaja na nivoju testiranja enot oz. modulov (Naik & Tripathy, 2008, str. 20–21, 42; Baresi & Pezzè, 2006; Cai et al., 2005, str. 552–579).

Testiranje črne skrinjice imenujemo tudi testiranje funkcionalnosti programske opreme. Pri tej vrsti testiranja nimamo dostopa do internih detajlov programske opreme, zato je le-ta predstavljena kot črna škatla, torej ne vemo, kaj se v njej dogaja. Testnega inženirja zanimajo samo vhodni in vidni izhodni podatki. Pri tem so uporabljeni vhodni podatki izbrani iz specifikacije zahtev in lastnosti vhodov in izhodov programa. To vrsto testiranja uporabljamo pri testiranju enot oz. modulov in pri testiranju sistema (Naik & Tripathy, 2008, str. 20–21, 42; Baresi & Pezzè, 2006; Cai et al., 2005, str. 552–579).

Ideja testiranja funkcionalnosti in strukturnega testiranja programerjem in testnim inženirjem ne daje izbire, ali bodo testni primeri oblikovani na osnovi izvorne kode ali na osnovi specifikacije zahtev programske opreme. Izbire nimajo, ker je v življenjskem ciklu razvoja programske opreme

potrebno v različnih časovnih obdobjih razvoja uporabiti različno vrsto testiranja. Tudi strokovnjaki so mnenja, da samo ena vrsta od navedenih dveh ni dovolj za uspešno odkrivanje napak, ampak je potrebna uporaba obeh, saj se medsebojno dopolnjujeta (Naik & Tripathy, 2008, str. 20–21, 42).

4.4.9 Samodejno testiranje

Testiranje je časovno in finančno zahteven proces. Zaradi vse večjega obsega programske opreme in zaradi stalnega nadgrajevanja le-te je lahko ročno testiranje ozko grlo v življenjskem ciklu razvoja programske opreme. V te namene se uvaja avtomatsko testiranje, ki je v primerjavi z ročnim hitrejše in natančnejše (Naik & Tripathy, 2008, str. 391–402). Samodejno testiranje predstavlja avtomatsko izvajanje testnih primerov, ki jih požene programer ali testni inženir. Potrebno se je zavedati, da mora še vedno nekdo pripraviti testne primere, ki se samodejno izvajajo. Poleg tega je potrebno tudi preučiti rezultate, ki jih testiranje avtomatsko ustvari.

Samodejno testiranje skrajša čas testiranja in življenjski cikel razvoja, posledično zmanjšuje tudi stroške testiranja in razvoja na dolgi rok. Olajša razhroščevanje, saj je ta način testiranja nedvoumen in podroben. Poleg tega omogoča testiranje določenih komponent, ki se jih ročno ne da testirati, na primer test obremenitve sistema. Ta način testiranja olajšuje izvajanje večje količine testnih primerov hkrati, zato je v veliko pomoč tudi pri testu regresije (Naik & Tripathy, 2008, str. 391–402).

Tak način testiranja predstavlja kompleksen in drag razvoj programske opreme, ki to testiranje izvaja. Zaradi različnih potreb in ciljev testiranja podjetja to programsko opremo najraje razvijajo sama. Priporočljivo je, da je njen razvoj voden. Tudi tu je potrebno razhroščevanje. Strokovnjaki s tega področja priporočajo, da se programska oprema za samodejno testiranje razvija v obliki modulov ali enot, kar omogoča parcialno samodejno testiranje in ponovno uporabo določenih modulov pri drugih projektih (Naik & Tripathy, 2008, str. 391–402).

Opisana vrsta testiranja nudi številne prednosti, kot so: povečana produktivnost testerjev, boljša pokritost testov regresije, krajše trajanje testnih faz, nižji stroški vzdrževanja programske opreme in povečana učinkovitost testnih primerov (Naik & Tripathy, 2008, str. 391–402).

Skozi preučevano literaturo sem spoznala, da samodejno testiranje ne more v celoti nadomestiti ročnega. Z obravnavanim testiranjem ne moremo zaznati določenih problemov, ki jih tester z ročnim testiranjem odkrije z lahkoto. Nekatera področja testiranja, kot so uporabnost, robustnost, kompatibilnost, navadno niso primerna za avtomatizacijo. V večini primerov je za avtomatizacijo primernih 50 % vseh testnih primerov sistema (Naik & Tripathy, 2008, str. 391–402).

5 MODELI ZRELOSTI

Eden od pristopov izboljševanja managementa kakovosti projekta in razvoja programske opreme je uporaba modela zrelosti (angl. *maturity model*). Modeli predstavljajo nekakšen okvir za pomoč podjetju pri izboljševanju svojih procesov in sistemov. Lahko bi rekli, da opisujejo evolucijsko pot

k bolj organiziranemu in sistematično bolj zrelemu procesu.

Večina modelov zrelosti ima pet ravni zrelosti. Prva raven opisuje karakteristike najmanj zrelega podjetja, peta raven pa karakteristike najbolj zrelega podjetja. Popularni modeli zrelosti so: model uvajanja funkcij kakovosti programske opreme (angl. *Software Quality Function Deployment Model –SQFD*), model zrelostnih stopenj (angl. *Capability Maturity Model – CMM*), model integriranih zrelostnih stopenj (angl. *Capability Maturity Model Integration – CMMI*), modeli zrelosti projektnega managementa (angl. *Project Management Maturity Models – OPM3*) itd. (Naik & Tripathy, 2008, str. 546–547; Schwalbe, 2009, str. 322–325).

5.1 Model uvajanja funkcij kakovosti programske opreme (SQFD)

Model se osredotoča na definiranje zahtev uporabnikov in planiranje projekta razvoja programske opreme. Razvili so ga leta 1986, da bi bil gonilna sila za celovito obvladovanje kakovosti (angl. *Total Quality Management – TQM*). Rezultat uporabe tega modela je nabor merljivih tehničnih specifikacij produkta in njihovih prioritet. Posledično se lahko v podjetju, ki uporablja in izvršuje naloge tega modela, poviša produktivnost in lažje izpolni zahteve uporabnikov oz. naročnika (Schwalbe, 2007, str. 340).

5.2 Model zrelostnih stopenj (CMM)

Ta model predstavlja ogrodje za ocenjevanje kakovosti in izboljševanje procesov. Model zrelostnih stopenj ima pet ravni zrelosti. Ko se podjetje premakne iz ene ravni na drugo, to pomeni izboljšano izvajanje kakovosti razvoja programske opreme in nižje stroške, kar posledično izraža večjo učinkovitost (Subramanian, Jiang, & Klein, 2007, str. 616–627; Naik & Tripathy, 2008, str. 548–554). Ravni zrelosti so (Subramanian et al., 2007, str. 616–627; Naik & Tripathy, 2008, str. 548–554):

- **1. raven: Začetna raven (angl. *initial*)**
  Na tej ravni potek razvoja programske opreme ne sledi nobenemu modelu procesov in tudi planiranje v veliki meri ni prisotno. Gre za izvajanje rešitev ad hoc, kar posledično lahko privede tudi do kaosa.
- **2. raven: Ponovljivost (angl. *repeatable*)**
  Na drugi ravni se vzpostavi osnovno projektno vodenje za sledenje stroškom, urnikom in funkcionalnosti. Z zagotavljanjem potrebne discipline izvajanja procesov lahko uspešne prakse iz preteklosti ponovimo na projektih s podobnimi lastnostmi tudi v prihodnosti. Če lahko procese iz preteklosti, ki so bili uspešni, ponovimo, smo izpolnili cilj te ravni.
- **3. raven: Definiranost (angl. *defined*)**
  Na tej ravni igra ključno vlogo dokumentacija. Procesi povezani z managementom projekta in razvojnem aktivnosti programske opreme so dokumentirane, pregledane, standardizirane in integrirane v procese podjetja oz. organizacije. Beležita se funkcionalnost in z njo povezana kakovost. Spremlja se tudi stroške in urnike ter zagotovi, da se le-ti gibljejo znotraj nadzora.
- **4. raven: Upravljanje (angl. *managed*)**

Ključno vlogo imajo meritve, ki si prizadevajo za zbiranje in analiziranje produkta in procesov podjetja. Meritve se uporabljajo za pridobivanje vpogleda v kvantitativno kakovost produkta in procesov. Ko meritve pokažejo prekoračitev, se sprožijo korektivni ukrepi.

- **5. raven: Optimiziranje (angl. *optimizing*)**
  Podjetja, ki dosežejo to raven, stremijo k nenehnemu izboljševanju poslovnih procesov. To dosežemo v dveh korakih, in sicer z opazovanjem učinkov procesov na podlagi merjenja nekaterih ključnih meritev kakovosti, stroškov in časa razvoja programske opreme; drugi korak predstavlja učinke sprememb na procesu z uvajanjem novih tehnik, metod, orodij in strategij.

Po razvoju in uspešni uporabi modelov zrelostnih stopenj na področju programske opreme, ki ga poznamo pod imenom model zrelostnih stopenj programske opreme (angl. *Capability Maturity Model for Software – CMM-SW*), so se razvili modeli zrelostnih stopenj tudi za druga področja. Modeli so bili razviti za: sistemski inženiring (CMM –SE), integracijski razvoj produkta (IPD – CMM), združenje elektronske industrije 731, nakup programske opreme in področje ravnanja z ljudmi in z dobavitelji (CMM – SS) (Schwalbe, 2007, str. 341–342).

5.3 Model integriranih zrelostnih stopenj (CMMI)

Gre za pristop procesnega izboljšanja, ki podjetju omogoča učinkovito izvajanje procesov. Lahko se ga uporablja za vodilo izboljševanja procesov skozi projekt, oddelek in celotno podjetje (Schwalbe, 2009, str. 343). Model integriranih zrelostnih stopenj pomaga pri integraciji organizacijskih funkcij oz. funkcij podjetja, določanju ciljev in prioritet izboljšav procesa, daje napotke za procesno kakovost in zagotavlja izhodiščno točko za ocenjevanje trenutnih procesov. Ta model vključuje informacije iz modelov: CMM – SW, IPD – CMM, CMM – SE in CMM – SS (Naik & Tripathy, 2008, str. 554–555).

5.4 Zrelostni model managementa projektov (OPM3)

Leta 2003 je inštitut za projektni management (angl. *Project Management Institute – PMI*) v okviru razvojnega programa standardov (angl. *Standards Development Program*) objavil zrelostni model organizacijskega managementa projektov (angl. *Organisational Project Management Maturity Model – OPM3*). Model je osnovan na podlagi tržne raziskave, ki je bila izpolnjena s strani več kot 30.000 profesionalnih projektnih managerjev in vključuje 180 najboljših praks ter 2.400 rezultatov, zmogljivosti in ključnih kazalnikov uspešnosti. John Schlichter, programski direktor OPM3, meni, da bo standard pomagal podjetju oceniti in izboljšati zmogljivosti projektnega managementa in tudi zmogljivosti, ki so potrebne za doseganje strategije podjetja skozi izvajanje projektov. Standard bo zrelostni model projektnega managementa, ki bo določal standard odličnosti projekta, programa in portfelja managementa najboljših praks ter razlago potrebnih zmogljivosti za doseganje le-teh (Schwalbe, 2007, str. 342–343).

6 STANDARDI KAKOVOSTI

Danes se med podjetji in njihovimi proizvodi na trgu pojavlja vse večja konkurenca. Za doseganje poslovne odličnosti morajo podjetja svoje procese obravnavati povezano in vanje vključiti sistem

kakovosti. To dosežejo tako, da uvedejo poslovne standarde kakovosti, s katerimi želijo na sistematičen način doseči konkurenčno prednost, višjo produktivnost in dobičkonosnost. Za dosego takega cilja morajo sodelovati vsi zaposleni na vseh ravneh podjetja. V ta namen se trudijo čim natančneje opredeliti in hkrati zagotoviti doseganje zastavljenih standardov. Standardi kakovosti ponudijo podjetju okvir, znotraj katerega ima možnost iskati slabosti in izboljšave v svojih procesih.

Standard je dokument, ki navaja splošna in večkrat uporabna pravila, navodila ali značilnosti proizvodov, storitev ali z njimi povezanih procesov in proizvodnih postopkov in katerega upoštevanje ni obvezno (Zakon o standardizaciji, 1995). Standardi so se razvili zaradi potrebe po enotni opredelitvi pojmov, ki so potrebni za sodelovanje med kupci in proizvajalci. Vsak standard ima svoj namen, ki je ločen glede na tip proizvodnega procesa. Od namena je odvisen obseg zahtev standardov. Splošna in dosledna uporaba standardov se pokaže pri nastopih na trgih kot velika prednost. Z zadovoljitvijo večine zahtev standardov dosežemo kakovostno proizvodnjo svojih izdelkov (Božič, 2004, str. 6).

Standardi kakovosti določajo oblikovanje politike podjetja za posamezna področja. Eden izmed načinov doseganja zastavljenih ciljev podjetja je uvajanje standardov kakovosti. Pri teh je zelo pomembno, da so vsi sistemi znotraj podjetja skladni s standardi, bistvena pa je njihova vključitev v vse sfere podjetja ter njihova oživitev v podjetju. Šele ko podjetje to doseže, izpolnjuje zastavljene cilje (Božič, 2004, str. 3–4).

S pomočjo standardov kakovosti podjetja oskrbujejo kupce s kakovostnimi, dobrimi proizvodi in storitvami. Najbolj poznani standardi pa so gotovo ISO in IEEE, ki jih bom obravnavala v nadaljevanju. Osredotočila se bom predvsem na tiste ISO standarde, ki so povezani z informacijsko tehnologijo.

6.1 ISO standardi

Mednarodna organizacija za standardizacijo (angl. *International Standard Organisation – ISO*) je bila ustanovljena leta 1946 v Ženevi v Švici (Naik & Tripathy, 2008, str. 534–535). Danes združuje 157 nacionalnih organov za standarde, ki v tej organizaciji predstavljajo interese držav članic, iz katerih prihajajo. Sestavljajo jo tehnični odbori, pododbori in delovne skupine, v katerih člani razvijajo in pripravljajo mednarodne standarde. Poleg nacionalnih organov za standarde lahko pri razvoju standardov v okviru ISO sodelujejo tudi druge mednarodne organizacije, ki imajo status povezanih članic. Nacionalni organi za standarde so organi priznani na nacionalni ravni, da kot nacionalni člani zastopajo interese svojih držav v ustreznih mednarodnih in evropskih organizacijah za standardizacijo. Slovenijo v mednarodnih in evropskih organizacijah za standardizacijo zastopa Slovenski inštitut za standardizacijo (SIST, 2011).

6.1.1 ISO 9000

Standardi družine ISO 9000 so zbirka veljavnih mednarodnih standardov, tehničnih specifikacij, tehničnih poročil, priročnikov in internetnih dokumentov o vodenju kakovosti. V začetku so bili zasnovani na predhodno izdanih standardih, ki so veljali predvsem za letalsko, vojaško, vesoljsko in

avtomobilsko industrijo. Družina standardov ISO 9000 predstavlja mednarodno soglasje o dobrih praksah managementa kakovosti. Sestavljena je iz standardov in smernic, ki se nanašajo na sisteme managementa kakovosti in s tem povezanih podpornih standardov (ISO 9000, 2011).

Družina ISO 9000 obsega 24 standardov ter predstavlja mednarodna merila in smernice za obvladovanje kakovosti. Trije od teh, ISO 9001, ISO 9002 in ISO 9003, so postali vodič podjetjem, ki se želijo certificirati. Certifikati kakovosti so zahteva strank in kupcev; zagotavljajo jim kakovosten izdelek oziroma storitev (Šircelj, 2006, str. 11).

Standardi ISO 9000 se od časa do časa pregledujejo in posodabljajo, navadno vsakih 5–8 let. Zadnja posodobitev je bila leta 2005. Večina člankov, knjig in ostalih virov opisujejo in predstavljajo ISO 9000 standarde iz leta 2000, ki jih zapišemo kot ISO 9000:2000 (predzadnja različica). Ti standardi se lahko uvedejo na vseh področjih, kjer je pri izdelavi produkta prisoten človeški faktor, tj. od proizvodnje začimb pa do razvoja programske opreme. Standardi ISO 9000:2000 so sestavljeni iz treh komponent (Naik & Tripathy, 2008, str. 535):

- ISO 9000: Osnove in slovar
- ISO 9001: Zahteve
- ISO 9004: Smernice za izvajanje izboljšav

Zaradi posodabljanja standardov vsakih nekaj let so na primer ISO 9002 in ISO 9003 izključili iz različice ISO 9000:2000, čeprav sta bila še del ISO 9000:1994. **ISO 9002** je obravnaval kakovost sistema za zagotavljanje kakovosti v proizvodnji in pri namestitvi. **ISO 9003** pa je obravnaval zagotavljanje kakovosti sistema pri končnem pregledu in testiranju (Naik & Tripathy, 2008, str. 535).

ISO 9000: Osnove in slovar temeljijo na osmih načelih (Naik & Tripathy, 2008, str. 535–537; ISO 9000, 2011):

- **1. načelo: Osredotočenost na naročnika (angl. *Customer Fucus*)**
  Podjetje teži k temu, da s svojim produktom zadovolji vsem zahtevam strank in preseže njihova pričakovanja. Za dosego tega cilja mora podjetje nameniti veliko časa ocenjevanju potreb, zahtev in pričakovanj potencialnih strank. Z zbranimi podatki iz faze ocenjevanja morajo le-te preučiti in analizirati ter zaključke vpeljati, na primer v proces razvoja programske opreme.
- **2. načelo: Vodstvo (angl. *Leadership*)**
  Naloga vodstva je zagotavljanje pozitivnega okolja, kjer lahko vsak izrazi svoje mnenje, in vse do tega, da nudijo podporo svojim podrejenim. Za uspešno managiranje je potrebno sodelovanje vseh zaposlenih, ki stremijo k istemu cilju. Da lahko podjetje uspešno posluje, so vodilni dolžni za zaposlene vzpostaviti primerno komunikacijo, razumevanje, motivacijo in stabilnost okolja.
- **3. načelo: Vključenost ljudi (angl. *Involvement of People*)**
  Na splošno velja, da podjetje predstavljajo ljudje, ki opravljajo določene naloge. Te so odvisne od posameznikov in njihovih spretnosti, znanj in izkušenj. V primernem okolju imajo posamezniki pri opravljanju svojih nalog možnost osebnega in tehničnega razvoja ter ustvarjalnosti. V takih pogojih ima podjetje veliko več možnosti biti uspešno in zadostiti zastavljenim standardom.

- **4. načelo: Procesni pristop (angl. *Process Approach*)**
  Proces predstavlja zaporedje aktivnosti, ki iz vhodov tvorijo izhode. S pomočjo planiranja, organiziranja in časovnega razporejanja aktivnosti podjetja lahko definirajo, ponavljajo ali merijo procese. Na ta način postane učinkovitejše in uspešnejše. Zelo pomembno je tudi, da ne pozabimo na stalne izboljšave izvajanja procesov.
- **5. načelo: Sistematičen pristop managementa (angl. *System Approach to Management*)**
  Celotni sistem sestavljajo različni procesi, ki so v različnih medsebojnih razmerjih. Velikokrat en proces vpliva na več procesov z različnih področij. V ta namen se podjetja trudijo opredeljevati in analizirati svoje procese in razmerja med njimi. S tem lahko preprečijo negativne posledice in hkrati povečajo učinkovitost in uspešnost svojega poslovanja.
- **6. načelo: Stalne izboljšave (angl. *Continual Improvement*)**
  S pomočjo sledenja procesom v rednih časovnih presledkih lahko odkrijemo različne izboljšave procesov, produktov, aktivnosti itd. Z uvajanjem izboljšav podjetje napreduje in na dolgi rok znižuje stroške, na primer razvoj programske opreme in predvsem njenega vzdrževanja. Ker si vsa podjetja prizadevajo znižati stroške in porabiti čim manj časa za vzdrževanje že razvitih produktov, morajo oblikovati primerne politike za zagotavljanje kakovosti.
- **7. načelo: Dejanski pristop k procesu odločanja (angl. *Factual Approach to Decision Making*)**
  Odločitve sprejemamo na podlagi dejstev, izkušenj in intuicije. Dejstva lahko zberemo z jasnimi meritvami procesa. Na osnovi meritev, izkušenj in intuicije se podjetje lažje odloči o svojem nadaljnjem delovanju, uvajanju izboljšav (katere izboljšave uvesti), sprejemanju odločitev itd.
- **8. načelo: Vzajemne koristi v odnosu z dobavitelji (angl. *Mutually Beneficial Supplier Relationships*)**
  Redko kdaj podjetje samo proizvaja vse dele, ki jih mora uporabiti v procesu izdelave svojih produktov. V tem procesu je pomembno ocenjevanje kakovosti, ustreznosti in ostalih lastnosti delov produkta, ki jih podjetje prejme od dobavitelja. Podjetje ima cilj, da uspe najti dobrega dobavitelja, s katerim lahko sodeluje in zagotovi izdelavo predvidenega končnega izdelka.

Namen serije ISO 9000 je pomagati organizacijam oz. podjetjem pri iskanju in uvajanju izboljšav, ki se kažejo v večji uspešnosti, boljši učinkovitosti, preprečevanju napak in nižanju tveganja ali katerikoli kombinaciji naštetih izboljšav. Eden izmed ciljev revizije standardov ISO 9000 je prilagoditev količine in razčlenjenosti zahtevane dokumentacije (Potkonjak, 2011).

6.1.2 ISO 9001

Zadnja posodobitev tega standarda se je zgodila leta 2008 (ISO 9001:2008). Večina pridobljene literature se sklicuje na različico iz leta 2000 (ISO 9001:2000), zato bom v nadaljevanju predstavila to tudi (ISO 9000, 2011). Med obema ni bistvene razlike. V novejši različici so dodana dodatna pojasnila za obstoječe zahteve ter nekatere spremembe, ki se navezujejo na ostale ISO standarde (Naik & Tripathy, 2008, str. 537). Standard ISO 9001:2000 specificira zahteve za sisteme kakovosti managementa v primerih, ko mora podjetje dokazati svojo sposobnost, da dobavlja proizvode, ki izpolnjujejo zahteve odjemalcev in zahteve ustreznih regulativ (Potkonjak, 2011).

ISO 9001:2000 obsega osem delov, pri čemer se prvih treh delov ne vključuje v izvajanje in sistem managementa kakovosti. Prvi trije deli s področja zahtev so: obseg, referenčni normativi, pogoji in opredelitve ISO 9001. Ostali pomembnejši deli pa obsegajo dele zahtev 4–8 po ISO 9001:2000, in sicer (Naik & Tripathy, 2008, str. 537–542; ISO, 2011; Toplak & Urbajs, 2003):

- **4. del: Sistem managementa kakovosti (angl. *Quality Management System – QMS*)**
  Pri tem konceptu se prepletata politika in cilji kakovosti. Posamezne aktivnosti za realizacijo so opredeljene v obliki interakcije procesov, ki zagotavljajo kakovost. Pri sistemu managementa kakovosti so pomembne tudi meritve in analize procesov, ki iščejo področja za izboljšave. Ustrezna dokumentiranost je zelo pomemben del QMS. Vse podrobnosti medsebojnega delovanja procesov morajo biti dokumentirane. Čista dokumentacija je ključ do razumevanja, kako en proces vpliva na drugega.
- **5. del: Odgovornost managementa (angl. *Management Responsibility*)**
  Management si mora prizadevati, da se celotno podjetje zaveda ciljev in politik zastavljene kakovosti ter da se zaposleni venomer trudijo dosegati cilje in izvajati ustrezno politiko kvalitete. Če management uspe to doseči, ima podjetje bistveno večjo možnost doseganja zastavljene kakovosti.
- **6. del: Management virov (angl. *Resource management*)**
  Viri so bistvenega pomena za doseganje ciljev in politik v podjetju. Za izvajanje posamezne aktivnosti v procesih je potrebno dodeliti ustrezno vrsto in količino virov. Poznamo različne vrste virov, glavni pa so: osebje, oprema, orodje, finančni viri in stavbe (prostori). Značilno je, da različne vire nadzorujejo in kontrolirajo različni oddelki oz. zaposleni v podjetju. Pri managementu virov je pomembno, da le-te porazdelimo glede na potrebe posameznega projekta, da se ga lahko v normalnih razmerah izvaja nemoteno.
- **7. del: Realizacija produkta (angl. *Product Realisation*)**
  Ta del obravnava procese, ki pretvarjajo zahteve kupcev v produkte oz. izdelke. Ključni elementi realizacije dela so na primer priprava načrta produkta, zajemanje zahtev naročnika ali kupca, preučevanje zahtev, vzpostavitev mehanizma in infrastrukture za nadzor izdelave produkta, spremljanje spreminjanja potreb itd.
- **8. del: Meritve, analize in izboljšave (angl. *Measurement, analysis and improvement*)**
  Ta del obravnava meritve, analize izmerjenih podatkov in nenehne izboljšave. Merjenje kazalnikov uspešnosti posameznega procesa pomaga ugotoviti, kako dobro se proces izvaja. Če ugotovimo, da je pod želeno ravnjo, poskušamo izvajanje procesa izboljšati s korektivnimi ukrepi, ki jih sprejmemo v fazi planiranja. Meritve so zaželene, saj v nasprotnem primeru ne moremo objektivno določiti in sprejeti izboljšav procesov.

Z izvajanje standarda ISO 9001:2000 podjetje pridobi prednosti, in sicer: prožen sistem managementa, spodbujanje nenehnega izboljševanja, izvajanje temelji na procesih, osredotočanje na zadovoljstvo odjemalca kot merilo za uspešnost, motivacija zaposlenih s skupnimi cilji ter sodelovanjem, izpolnjevanje zakonskih zahtev in regulativ, osredotočenost na učinkovito notranje komuniciranje, usmerjenost pozornosti k razpoložljivim virom, vključevanje celotnega podjetja (tudi management) v izvajanje itd. (Toplak & Urbajs, 2003; SIST, 2011; Potkonjak, 2011).

Uporaba QMS pomeni sistematično izvajanje različnih aktivnosti. Te aktivnosti so lahko na primer

določitev politike in ciljev kakovosti, določitev in uporaba meril za preverjanje kakovosti, določitev in specifikacija ključnih procesov, zmanjševanje neskladnosti in težnja po stalnih izboljšavah, primerjava dejanskih rezultatov s planiranimi izhodi itd. (Toplak & Urbajs, 2003; SIST, 2011; Potkonjak, 2011).

6.1.3 ISO 9004

Najnovejša različica tega standarda je izšla leta 2009. V splošnem ISO 9004 podaja smernice, ki se nanašajo na učinkovitost in tudi na uspešnost sistema managementa kakovosti. Namen tega standarda je izboljševanje delovanja organizacije ter zadovoljstva odjemalcev in drugih zainteresiranih strani (Potkonjak, 2011; What is ISO 9004, 2011).

Večina projektov je kreditiranih na enem od zgoraj omenjenih standardov. Hkrati pomeni, da nekdo oz. eno izmed akreditiranih teles izvaja kontrolo nad organizacijo in preverja, ali je vse skladno z zahtevami (Kousholt, 2007, str. 57–63).

ISO 9004 zajema management, strategijo, sistem managementa, vire in procese za opredelitev prednosti in slabosti ter možnosti za izboljšave in inovacije. Zagotavlja višjo osredotočenost na upravljanje kakovosti kot ISO 9001. Obravnava potrebe in pričakovanja vseh ustreznih zainteresiranih strani. Določa tudi smernice za sistematično in stalno izboljševanje kakovosti. Lahko se uporablja skupaj z ISO 9001 in drugimi standardi ali samostojno (What is ISO 9004, 2011).

6.1.4 ISO 9126

Trenutno je to eden izmed najbolj razširjenih standardov kakovosti. Gre za standard, ki vrednoti programsko opremo. Obsega kakovost meritev in modelov. Zagotavlja opredelitev značilnosti in z njimi povezane postopke ocenjevanja kakovosti. To se uporablja pri določanju zahtev za ocenjevanje in ovrednotenje kakovosti programske opreme in produktov v njihovem življenjskem ciklu (Botrlla at al., 2011; ISO 9126, 2011).

ISO 9126 se osredotoča na šest lastnosti, ki opisujejo kakovost programske opreme z minimalnim prekrivanjem med seboj. Posamezne lastnosti se delijo na podlastnosti. V spodnji tabeli je prikazanih vseh šest glavnih lastnosti, njihove podlastnosti ter opis pomena le-teh (Botrlla et al., 2011; ISO 9126, 2011; Naik & Tripathy, 2008, str. 530–534).

Slika 7: Lastnosti podlastnosti in razlaga podlastnosti standarda ISO 9126

Lastnosti	Podlastnosti	Opredelitev podlastnosti
funkcionalnost	primernost	Zmožnost programske opreme, da ponuja primerno funkcionalnost.
	natačnost	Predstavlja zmožnost pravilnega izvajanja funkcij programske opreme.
	vezljivost	Zmožnost delovanja programske opreme z drugimi komponentami ali sistemi.
	varnost	Sposobnost preprečevanja neavtoriziranih dostopov do funkcij programske opreme. Ni omogočeno spreminjanje in branje zaupnih podatkov.
zanesljivost	zrelost	Predstavlja pogostost odpovedi programske opreme.
	toleranca napak	Sposobnost programske opreme, da vzdržuje določeno raven delovanja v primeru napak.
	okrevanje	Sposobnost programske opreme, da ponovno vzpostavi in obnovi njeno zmogljivost v primeru napake.
uporabnost	razumljivost	Zmožnost, da lahko uporabnik razume delovanje in možnosti uporabe programske opreme.
	učljivost	Zmožnost, da se uporabnika lahko nauči rabiti programsko opremo.
	delovanje	Zmožnost izvajanja kontrole, nadzora in upravljanja programske opreme.
učinkovitost	poraba časa	Sposobnost, da zagotavlja ustrezen odzivni čas, obdelavo in pretočno stopnjo podatkov.
	uporaba virov	Sposobnost uporabe ustreznih virov v ustreznem času za izvedbo sproženih funkcij.
vzdrževalnost	možnost analiz	Sposobnost diagnosticiranja pomanjkljivosti in napak v programski opremi.
	spremenljivost	Zmožnost programske opreme, da lahko določene enote v njej spremenimo ob normalnem delovanju ostalih delov.
	stabilnost	Zmožnost, da minimizira nepredvidene odzive na spremembe v programski opremi.
	možnost testiranja	Omogočanje testiranja vseh enot programske opreme.
prenosljivost	prilagodljivost	Sposobnost programske opreme, da se prilagodi okolju uporabnikov brez uporabe dodatnih ukrepov ali sredstev.
	možnost namestitve	Možnost namestitve programske opreme v določeno okolje oz. sistem.
	skladnost	Sposobnost programske opreme, da deluje neodvisno v skupnem okolju s skupnimi viri z ostalo programsko opremo uporabnika.
	nadomestljivost	Možnost nadomestitve druge programske opreme z novo v istem okolju.

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

V nadaljevanju sledi obrazložitev še posameznih lastnosti iz zgornje tabele.

Funkcionalnost programske opreme vključuje nabor lastnosti, ki so zadolžene za omogočanje funkcij le-te in določenih lastnosti, v zgornji tabeli imenovanih podlastnosti. Funkcije so tisto, kar izpolnjuje navedene ali posredno izražene zahteve programske opreme (Botrlla et al., 2011; Naik &

Tripathy, 2008, str. 530–534; SQA, 2011).

Sledi ji **zanesljivost**, ki vključuje nabor lastnosti za zagotavljanje ravni delovanja programske opreme ob določenih pogojih in za predvideno časovno obdobje (Botrlla et al., 2011; Naik & Tripathy, 2008, str. 530–534; SQA 9126, 2011).

Naslednja lastnost je **uporabnost**, kjer se preverja, ali programska oprema služi svojemu namenu. Ta test navadno opravi končni uporabnik bodoče programske opreme, ki od nje pričakuje izpolnitev vseh navedenih zahtev in pričakovanj (Botrlla et al., 2011; Naik & Tripathy, 2008, str. 530–534; SQA, 2011).

Učinkovitost je lastnost, ki obravnava razmerje med delovanjem programske opreme in njeno porabo virov pri določenih pogojih. Najpogosteje se to odraža v obliki odzivnega časa aplikacije in količino porabe razpoložljivih virov (Botrlla et al., 2011; Naik & Tripathy, 2008, str. 530–534; SQA, 2011).

Vzdrževalnost mora upoštevati, da se bodo v prihodnosti na tej programski opremi potekale spremembe, kot so: popravki, izboljšave, prilagoditve, okoljske spremembe, spremembe v zahtevah in specifikaciji (Botrlla et al., 2011; Naik & Tripathy, 2008, str. 530–534; SQA 9126, 2011).

P**renosljivost** omogoča, da lahko programsko opremo vključi v različna delovna okolja ali prenaša iz enega v drugo okolje, pri čemer mora njena funkcionalnost ostati enaka (Botrlla et al., 2011; Naik & Tripathy, 2008, str. 530–534; SQA 9126, 2011).

6.2 IEEE standardi kakovosti

Poleg omenjenih ISO standardov poznamo tudi druge. Eni izmed pomembnejših na področju tehnologije so tudi IEEE standardi (angl. *Institute of Electronical and Electronics Engineers standards*).

IEEE je največja profesionalna organizacija usmerjena v inovacije napredne tehnologije in odličnosti za dobro družbe. Zasnovana je tako, da služi strokovnjakom na električnih, elektronskih in računalniških področjih in z njimi povezanimi področji znanosti in tehnologije, ki so vodilo moderne civilizacije. S časom je tehnologija napredovala in se širila na različna področja, zato so postali tudi strokovnjaki, kot so na primer razvijalci programske opreme, znanstveniki na področju računalništva, strokovnjaki informacijske tehnologije, fiziki, zdravniki in mnogi drugi del organizacije IEEE. Zaradi vse več področij, ki jih organizacija IEEE pokriva in vključuje pri oblikovanju standardov, svoje polno ime čedalje manjkrat uporablja. Polno ime tako uporablja le, kjer je to izrecno zahtevano, drugje pa se predstavlja s kratico IEEE (About IEEE, 2011).

Organizacija IEEE je nastala kot posledica združenja organizacij IRE (angl. *Institute of Radio Engineers*) in AIEE (angl. *American Institute of Electrical Engineers*) 1. januarja 1963 s sedežem v New Yorku. Institucija IRE je bila ustanovljena leta 1912 zaradi potreb strokovnjakov na področju

radijskih komunikacij. Institucija AIEE je bila ustanovljena spomladi 1884 z namenom podpore strokovnjakom razvijajočega se tehničnega in električnega področja in kot pomoč pri prizadevanjih za inovacije, ki stremijo k izboljšavam za človeštvo (History of IEEE, 2011).

Do začetka 21. stoletja je organizacija izdala preko 130 časopisov in revij, organizirala več kot 300 konferenc letno in v praksi uporabljala 900 aktivnih standardov. Do leta 2010 je imela več kot 395.000 članov v 160 državah sveta. S svojo svetovno mrežo geografskih enot, publikacij, spletnih storitev in konferenc IEEE ostaja največje svetovno tehnično strokovno združenje (History of IEEE, 2011).

Področje managementa kakovosti razvoja programske opreme zajema vrsto IEEE standardov. Za boljšo predstavo obsežnosti tega področja bom navedla le nekaj primerov teh standardov, saj je v okviru magistrskega dela nemogoče predstaviti vse standarde navezujoče na to področje. Pomembnejši IEEE standardi, katerih tematike sem se dotaknila v okviru magistrskega dela, so: Standardni slovar programske opreme inženirske terminologije (610.12 – 1990), Plan zagotavljanja kakovosti programske opreme (730 – 2002), Testiranje enot programske opreme (1008 – 1987), Verifikacija in validacija programske opreme (1012 – 2004), Pregledi in revizije programske opreme (1028 – 2008), Razvrstitev anomalij za programsko opremo (1044 – 2009), Management projektnega plana programske opreme (1058 – 1998), Dokumentacija uporabnikov programske opreme (1063 – 2001), Življenjski cikli procesov razvoja programske opreme (1074 – 2006), Vzdrževanje programske opreme (1219 – 1998), Informacijska tehnologija – Življenjski cikel procesov sistema in programske opreme – Ponovna uporaba procesov (1517 – 2010), Življenjski cikel procesov programske opreme – Management tveganja (1540 – 2001) itd. (IEEE Standard association, 2011).

IEEE standardi se stalno posodabljajo, ko so pri posameznem standardu potrebne razširitve, popravki ali druge spremembe. Nekateri ohranjajo svoja imena in identifikacijske številke, medtem ko se drugi združujejo s standardi drugih organizacij za standardizacijo, kot je na primer ISO organizacija. V tem primeru pridobi standard drugo ime in identifikacijsko številko.

6.2.1 Testiranje enot programske opreme

Standard, ki določa testiranje enot programske opreme, ima identifikacijsko številko 1008 in je bil sprejet leta 1987. Vsak standard je bil sprejet zaradi ciljev, ki jih z njim želimo doseči. Primarni cilj standarda je določitev standardnega pristopa k testiranju enot programske opreme, da se lahko uporablja kot dobra praksa pri njenem razvoju. Drugi cilj je opisati koncepte in predpostavke testiranj programske opreme, na katerih temelji standardni pristop. Tretji cilj je zagotovitev smernic in virov informacij za pomoč pri izvajanju in uporabi standardnega pristopa pri testiranju enot (IEEE Standard association: 1008 – 1987, 2011).

6.2.2 Verifikacija in validacija programske opreme

Standard ima identifikacijsko številko 1012 in je bil nazadnje posodobljen leta 2004. Procesi verifikacije in validacije programske opreme potekajo zato, da bi ugotovili, ali je produkt, ki smo ga

razvili, v skladu s podanimi zahtevami in izpolnjuje pričakovanja ter potrebe uporabnikov oz. naročnika. Zahtev procesov življenjskega cikla razvoja programske opreme so različne glede na nivo integritete programske opreme enot (IEEE Standard association: 1012 – 2004, 2011).

Obseg procesov standarda zajema: strojno opremo, vmesnike, programsko opremo in sisteme, ki temeljijo na programski opremi. Standard velja za programsko opremo, ki se razvija, vzdržuje ali ponovno uporabi. Izraz programska oprema v tem primeru vključuje tudi mikro kodo, dokumentacijo in vgrajene programe (programi, ki tečejo na mikroprocesorjih) (IEEE Standard association: 1012 – 2004, 2011).

Procesi verifikacije in validacije programske opreme vključujejo analizo, ovrednotenje, pregled, nadzor, ocenjevanje in testiranje (IEEE Standard association: 1012 – 2004, 2011).

6.2.3 Pregledi in revizije programske opreme

Standard ima identifikacijsko številko 1028 in je bil nazadnje posodobljen leta 2008. Standard zajema pet tipov pregledov in revizij programske opreme in postopke, ki so potrebni za izvedbo posameznega tipa. Tipi vključujejo management pregledov, inšpekcijske preglede, tehnične preglede, revizije in kažipote. Standard se ukvarja samo s pregledi in z revizijami, medtem ko postopki za ugotavljanje potrebnosti teh in razporeditev rezultatov niso opredeljeni (IEEE Standard association: 1028 – 2008, 2011).

6.2.4 Razvrstitev anomalij za programsko opremo

Standard ima identifikacijsko številko 1044 in je bil nazadnje posodobljen leta 2009. Predvideva enoten pristop za razvrstitev anomalij programske opreme, ne glede na to, kdaj so nastale oziroma kdaj se z njimi srečamo v projektu, produktu ali v življenjskem ciklu sistema. Razvrstitev teh podatkov se lahko uporablja za različne namene, tudi v analizi vzorčne napake, pri projektnem vodenju in izboljševanju programske opreme (IEEE Standard association: 1044 – 2009, 2011).

7 PRIMER PREVERJANJA KAKOVOSTI S TESTIRANJEM: PROGRAMSKA OPREMA ZA VISOKOFREKVENČNO TRGOVANJE

V nadaljevanju bom predstavila pomembnost preverjanja kakovosti s testiranjem na fiktivnem primeru programske opreme za visokofrekvečno trgovanje z vrednostnimi papirji. Primer bo zajemal predstavitev produkta in testiranje po ravneh, ki so opisane v poglavju 4.4.6.

Na podlagi tega primera bomo lahko spoznali, kako pomembno je testiranje in kakšne so lahko posledice, če ga ne izvedemo primerno in dovolj natančno. Obenem bom tudi predstavila posamezne vidike testiranja programske kode, ki bi preprečili nastop težav, s katerimi so se srečevali avtorji.

Pri vsaki novi programski opremi je ključnega pomena skrbno testiranje le-te, da zagotovimo zadovoljstvo naročnika in izvajalca. Zelo pomembno je, da se že ob začetku takšnih projektov zavedamo, da bo testiranje zahtevalo mnogo delovnih ur in razmišljanja, kako naj se izpelje. Velikokrat se podjetja, ki se ukvarjajo z razvojem programske opreme, otepajo testiranja. Imenujejo ga tudi »zapravljanje časa«. Pomembno je pravilno in kakovostno testiranje, ki omogoči pravočasno odkritje napak in pomanjkljivosti programske opreme. S tem zagotovimo, da je produkt skladen s strani naročnika potrjeno specifikacijo in da primerno delovanje produkta, preden je produkt v redni uporabi. Če testiranju ne namenimo dovolj časa, prihaja do velikih težav pred predajo produkta, ko napake prihajajo na dan. Kot vemo, so posamezne napake zelo kompleksne, njihovo odpravljanje pa ima pogosto za posledico zamujanje rokov oddaje.

7.1 Opis produkta

Predstaviti želim incident, ki se je zgodil v začetku avgusta 2012 na glavni ameriški borzi v New Yorku – NYSE. Ameriška borznoposredniška hiša Knight Capital je 1. 8. 2012 zagnala nov programski paket – Retail Liquidity Program za visokofrekvenčno trgovanje – imenovan tudi *algo*, *black-box* oz. *robo trading*, ki predstavlja novejši način trgovanja na borzi (Olds & Consulting, 2012; Silver-Greenberg, Popper, & De La Merced, 2012). Posamezna programska oprema v tem primeru prevzame vlogo borznega posrednika pri odločanju o posameznih sestavnih delih naročila, kot so: čas vložitve, nakupna oz. prodajna cena, količina in podobno. Ključna prednost takšnega načina trgovanja je v hitrosti odločanja, ki se meri v milisekundah. S tem je omogočeno izkoriščanje tudi najmanjših priložnosti za pozitivno arbitražo – izvede se nakup po nekoliko nižji ceni in takojšnjo prodajo po višji ceni.

Prvi tovrstni sistemi so v uporabi od sedemdesetih let dalje, čedalje bolj so se razširili v zadnjih desetih letih. V zadnjem času se take sisteme namešča v kletne prostore borznih hiš, da bi dosegli kar najmanjši časovni zamik pri sprejemanju podatkov. Obseg opisanih poslov je zelo velik – v ZDA predstavljajo taki posli že 75 % vseh poslov z delnicami. Podobno velja tudi za Evropo. Kot zanimivost lahko omenim, da sta letos (2012) v poletnem času podjetji Artic Fibre in Artic Cable prevzeli polaganje nove podmorske povezave (kabelska povezava) med Veliko Britanijo in Japonsko zato, da bi se časovni zamik v prometu med obema borzama lahko zmanjšal za 60 tisočink sekunde (z 230 na 170 milisekunde). Teh 30 % prihranka v času zanima predvsem velike investicijske hiše. Polaganje omenjene podmorske povezave bo stalo kar 1,5 milijarde dolarjev (Williams, 2011).

Knight Capital je s prvim avgustom začel z uporabo novega sistema za algoritmično trgovanje, ki je bil potreben zaradi sprememb na borzi. Na ta dan so prilagodili svoj vmesnik za poslovanje z večjimi strankami. Uvedba novega sistema je bila napovedana konec lanskega leta (2011), medtem ko je idejo odobril regulator letos (2012) maja (Chapman, 2012). Knight Capital je torej imel precej časa za prilagoditve svojega programa. Podjetju ni bilo treba pripraviti povsem novega programa, saj se je lahko večji del programske logike ohranilo. Potrebna je bila sprememba samo dela za vnašanje poslov.

7.2 Glavna težava programske opreme

Opisana programska oprema je šla v uporabo v sredo, 1. avgusta 2012 zjutraj. Zgodila se je velika katastrofa, saj program ni deloval pravilno. Knight Capital se je moral soočiti z visokimi izgubami zaradi nepravilnega delovanja enega pogojnega (angl. *if*) stavka, ki je zamešal prodajna in nakupna naročila. V nadaljevanju bom natančneje obrazložila, kaj se je zgodilo (Olds & Consulting, 2012; Silver-Greenberg et al., 2012).

Obrazložitev delovanja na hipotetičnem primeru
A = vrednostni papir
P_a,x = ponudba za prodajo določene količine vrednostnih papirjev A, po ceni X
N_a,y = ponudba za prodajo določene količine vrednostnih papirjev A, po ceni Y
T_a = trenutni tečaj vrednostnega papirja A

Če je T_a = 100, bi ob običajnem spoštovanju logike programska oprema morala avtomatsko sprejeti vsa nakupna naročila, ki so na voljo nad ceno 100 (npr.: P_a, 103). V navedenem primeru dobimo oz. lahko kupimo delnico za 100 in jo nato takoj prodamo kupcu za 103. V tem primeru bi s trgovanjem ene delnice dobili 3 enote dobička. Držati se moramo pravila, da je razlika med ponujeno in tržno ceno za sprejem nakupnega naročila pozitivna (velja torej P_a, 103 > T_a). Pri prodajnih količinah velja obratna logika; sistem sprejme tiste, kjer je nekdo ponudil prodajo delnice za manjšo vrednost od tečaja. V tem primeru mora biti razlika negativna (velja torej P_a, x – T_a).

Program je zaradi napake v kodi zamešal oba zapisana tipa naročila, obenem pa ni zamenjal logike za oba tipa naročila. Tako je pri ponudbah za prodajo pri na primer P_a, 110 pri T_a = 100 videl pozitivno tržno razliko s tržno ceno (110 – 100 > 0). Posledično je po kriterijih za ugodna *nakupna* naročila takoj sprejel. Posledica takega delovanja je bilo efektivno nakupovanje vseh vnesenih limitiranih prodajnih naročil. Poleg tega je sistem porezal vso ponudbo na trgu. Začarani krog se je odvijal 45 minut, preden so uspeli program zaustaviti (Silver-Greenberg et al., 2012).

Cena vseh delnic, s katerimi so trgovali – vseh je bilo 150 – je začela rasti. Vsakič je zrasla za delček odstotka, zato so ostali udeleženci na trgu začeli vnašati nova, dražja prodajna naročila – bili so navdušeni, da nekdo te delnice kupuje. Za 5 delnic je tečaj poskočil kar za 30 %. Posledica tega dogodka je bila, da je borza zamrznila in razveljavila trgovanje z omenjenimi delnicami. Za ostalih nekaj več kot 140 pa tega ni storila. Tako se je Knight Capitalu zgodilo, da je imel kup delnic v svojih rokah, ki jih ni potreboval za nikogar, zato jih je bil primoran prodati z visokim diskontom. Na koncu trgovalnega dneva so imeli izgube za približno 450 milijonov dolarjev – ocenili so ga kot enega največjih incidentov zaradi programske napake vseh časov. Knight Capital bi »potonil«, če njegovi lastniki ne bi uspeli pridobiti izrednega financiranja v višini 400 milijonov dolarjev. Vse to zaradi enega napačnega pogojnega stavka, ki je prodajna naročila označil za nakupna (Olds & Consulting, 2012; Silver-Greenberg et al., 2012).

Z dovolj testiranja se kaj takšnega ne bi smelo zgoditi. Zakaj je do tega prišlo, vedo samo odgovorni za incident v Knight Capitalu in njihov izvajalec. V nadaljevanju bom prikazala, kako bi

lahko z različnimi ravnmi testiranja sorazmerno enostavno preprečili opisano tragedijo.

7.3 Testiranje in njegove ravni

Test je programska metoda, ki izvrši nek drug del kode, pobere rezultat in se na nek način »pritoži«, če rezultat odstopa od pričakovanj. Navadno prej vzpostavimo delovno različico (angl. *mock*) produkcijskega okolja, podatkovne baze in drugih potrebnih sestavin.

Natančen način pisanja testov – sintaksa, uporabljene knjižnice, lokacija itd., so odvisni od programskega jezika do jezika, včasih tudi od uporabljene knjižnice za testiranje. Nekateri jeziki pridejo z že vgrajeno podporo ali sintakso za testiranje, pri drugih si to uredijo razvijalci sami.

Pri samem testiranju je zelo pomembna tudi kultura v podjetju, kjer se programska oprema razvija. Testi so del načrtovanja, ki marsikje spada v kategorijo »manj nujno« ali celo »ne nujno« fazo projekta. Velikokrat se tak odnos pojavlja, ker ni takojšnjega učinka testiranja. Pri veliko projektih se zato beleži popolna odsotnost testiranja, nepopolno testiranje, nepoganjanje ali zastarelost testov. Testi so učinkoviti le, če so vzdrževani. To zahteva določeno časovno in s tem denarno investicijo s strani razvijalske ekipe.

Na voljo imamo več ravni testiranja: test enot, test integracije, test sistema in test regresije, ki so podrobneje opisani v poglavju 4.4.6. Pri prvih treh vsaka stopnja preveri delovanje ustrezno večjega dela programskega paketa, medtem ko testiranje regresije poskrbi, da naknadne spremembe v kodi programa ne porušijo povezave v ostalih delih programa.

7.3.1 Test enot

Tipičen primer »enote« kode je posamezna metoda. Na primer metoda, ki sprejme odločitve za sprejem določenega borznega naročila. Koda algoritmičnega trgovalnega sistema v psevdokodi zgleda približno takole:

```
sub trguj (borza):
        while True:
                posel = borza.dobi_naslednji_posel()
                if posel.jeProdajni():
                        if posel.jeUgodni(trenutni_tecaj)
                                posel.izvedi()
                posel.porocaj()
```

Zapisan del kode – več kot očitna poenostavitev – ves čas se proces ponavlja čez vnešene posle in išče takšne prodaje oz. nakupe, ki so ugodni glede na trenutni tečaj.

V tem kontekstu bi lahko imeli po en test enot za vsako klicano funkcijo v funkciji trguj() ter še po en test enot za preverbo poslovne logike v funkciji. V nadaljevanju bom navedla nekaj primerov tipičnih testov za obravnavani primer:

- Postavi delovno različico (angl. *mock*) borza, vanj vstavi naročilo, nato preveri: če klic funkcije borza.dobi_naslednji_posel() in se vrne to naročilo le enkrat, funkcija deluje pravilno.
- Izdelaj prodajni posel in preveri: če funkcija jeProdajni() vrne, da je posel prodajni, potem funkcija deluje pravilno; enako velja tudi za nakupne posle.
- Vstavi prodajni posel in preveri: če funkcija jeUgodni() ustrezno ovrednoti smiselnost sprejema posla glede na parametre, potem funkcija deluje pravilno.
- Izvedi nekaj poslov in preveri: če so zanje izvedena ustrezna poročila, potem funkcija deluje pravilno.

Teste enot lahko pripravimo do poljubne stopnje granulacije oz. razdelanosti. Pokritost kode s testi je stvar odločitev in izkušenosti programerja in vodje projekta.

V konkretnem primeru, ki ga obravnava funkcija jeProdajni(), ni vračalo pravilnega rezultata testa, zato predvidevamo, da zanjo bodisi ni bilo testa bodisi test ni bil pognan bodisi test ni bil pravilen bodisi je bil test zastarel.

7.3.2 Test integracije

Izdelovalci poslovnega programa navadno ne dobijo neposrednega dostopa do podatkovne baze pri naročniku, ker je to zelo občutljivo tako glede varovanja poslovnih skrivnosti kot tudi zaradi varovanja osebnih podatkov. Navadno bo na novo napisani program deloval v navezi z obstoječim modulom ali podatkovno bazo naročnika, čemur pravimo »integracija«. Ko en podsistem poznamo, drugi pa deluje kot »črna skrinjica«, je še posebej pomembno, da v dovolj veliki meri stestiramo povezavo med njima. Enako velja tudi v primeru, če izvajalec pripravi oba podsistema sam; ali če je eden podsistem imel večji obseg; ali če na njih delata različni ekipi (kar je danes precej pogosta praksa); ali če se je eden od podsistemov zelo spreminjal. Velikokrat se zgodi, da napake najdemo šele pri robnih primerih.

Razumno je domnevati, da podatek o vnešenih poslih in o trenutnem tečaju pride z borze navadno na podlagi klicev omrežne funkcije ali z branjem toka (angl. *Stream of data*). To bi ustrezalo klicu funkcije borza.dobi_naslednji_posel(), ki mora interpretirati prejete podatke in iz njih izdelati objekt tipa Posel, ki ga razume sistem za algoritmično trgovanje.

Posledično lahko domnevamo, da je zaradi spremembe na strani borze ta sistem začel sprejemati napačne odločitve o tipu posla. Ker avgustovska sprememba zadeva ravno spremembo tega dela sistema, bi morali biti razvijalci še posebej temeljiti s tolmačenjem in preverjanjem informacij, ki jih dobijo od borze; po vsej verjetnosti jim to ni uspelo zaradi časovnega primeža ali kakšnega podobnega razloga. Skoraj zagotovo so namreč imeli dostop do testnega informacijskega sistema borze, kjer so bile spremembe na voljo vnaprej. Dober test integracije bi tako vanj vstavil nekaj naročil, jih dal nato prebrati v trgovalno aplikacijo ter jih primerjal za vse ključne značilnosti, med drugim tudi tip posla. Podobno bi lahko imeli spisane hevristične teste, ki bi zajeli večje število poslov in izločili možnost anomalij, npr. znatno število prodajnih naročil s prenizko ceno ali odsotnost nakupnih naročil ali kaj tretjega, kar se pri običajnem poslovanju pravzaprav ne bi smelo

zgoditi. Padli testi tega tipa bi programerjem jasno povedali, da je nekaj narobe. V najslabšem primeru bi ta alarm prišel takoj po splavitvi novega sistema, saj visokofrekvenčna programska oprema za trgovanje vidi tudi po več sto ali tisoč poslov vsako uro, morda celo sekundo. Test integracije zagotavlja, da se posamezne komponente v sistemu znajo pogovarjati med sabo ter da če ena komponenta izgubi to sposobnost zavoljo interne napake, ostale ne padejo kot domine.

7.3.3 Test sistema

Načrt za to testiranje bosta praviloma pripravila projektni manager s strani naročnika in izvajalca, in to v skladu z medsebojnim dogovorom. Test bo odlično zagotovilo, da sistem kot celota deluje in je tako v celoti skladen z dogovorjenimi zahtevami naročnika. Uspešno prestan sistemski test je tudi pogoj za prevzem sistema oz. dokončno plačilo izvedbe.

Test sistema bi moral biti sorazmerno obsežen. Na primer tak, ki bi v našem primeru testiral miselnost trgovalne strategije skozi daljše časovno obdobje ali vsaj nekaj deset poslov (naj še enkrat opozorim, da se krajše časovno obdobje v algoritmičnem trgovanju meri v milisekundah ali še manjših enotah). En sistemski test bi na primer pobral vsa jutranja naročila, jih ovrednotil in izvedel tako, da bi imel na koncu dneva čim večji dobiček. Dolgotrajna ali visoka izguba, še posebej kot v našem primeru slabe pol milijarde dolarjev, bi tukaj morala takoj sprožiti rdeči alarm. Če bi bil tak test pognan na borznem strežniku vsaj en dan pred prvim avgustom, bi programerji jasno videli, da je nekaj zelo narobe.

7.3.4 Regresijsko testiranje

Ključni napredek v zadnjih letih je uvedba t. i. sistema konstantnega testiranja (angl. *Continous integration*). Na strežnik, kjer se hrani koda (angl. *Code repository*), se namesti poseben program, ki čaka na novo prispelo kodo in nato vsako kopijo postavi v virtualnem okolju na novo ter izvede vse dotedanje teste. Če kateri od testov pade, se takoj razpošlje opozorilna e-sporočila vodji testiranja ter programerju, ki je v sistem vnesel zadnjo spremembo. Iz dnevnika tega testa je praviloma mogoče hitro ugotoviti, kaj je šlo narobe. Takšna testiranja zato izjemno pomagajo pri odpravi napak ob posodobitvi kode na strežniku.

Eden od takšnih sistemov je Travis CI, ki ga je mogoče najeti (po modelu software-as-a-service), za odprtokodne projekte je celo zastonj. Sicer bo držalo, da rezultate tovrstnega testiranje vidijo zgolj razvijalci sami, saj se izvaja zelo pogosto, obenem pa je močno odvisno predvsem od zadnjih nekaj sprememb. Projektni manager na strani naročnika ima ključno vlogo pri večjih spremembah, ki zadevajo zamenjavo nekega ključnega podsistema. Takrat se lahko zgodi – odvisno od kakovosti vmesnikov med podsistemi – da vsi ostali podsistemi, ki so bili odvisni od spremenjenega podsistema, začno delovati napačno. Projektni manager mora ob sprejemu vseh teh odločitev predvideti ustrezni čas in sredstva za razrešitev tovrstnih napak. Lahko se naroči na dnevne ali tedenske povzetke regresijskega testiranja, v katerih bodo naštete predvsem napake, ki kljub večkratni ponovitvi še niso razrešene.

7.4 Sklep za predstavljen primer

V sedmem poglavju sem testiranje enoznačno predstavila kot dobro idejo oz. kot nujnost, brez ki se je v končni fazi ne sme preskočiti. Če izvajalec preskoči fazo testiranja, se to šteje za malomarnost izvajalca. Testiranje pobere veliko časa, zato je tudi ustrezno višja končna vrednost produkta, lahko tudi za več kot 10 odstotkov. S testiranjem dejansko zamenjujemo dražje bodoče delo za cenejše in manj obsežno zdajšnje delo (pisanje testov).

Številni naročniki in izvajalci stremijo k rezanju cen. Posledica je navadno, da odpade najprej tisto, kar ni nujno potrebno sedaj (tukaj velikokrat izključimo testiranje). Če sprejmemo to idejo in s tem tudi idejo, da testiranje sedaj ni nujno potrebno, lahko povečamo možnost, da bo nekega nedelovnega dne (prazniki, sobota, nedelja) prišlo do hude napake, za računalnikom pa ne bo nikogar, ki bi to napako lahko odpravil. Poudariti je potrebno, da pri programski opremi niso vse napake tako drage, kot je stala napaka Knight Capital. Velikokrat večje napake odkrijejo, še preden gre programska oprema v redno uporabo.

SKLEP

V današnjem času se področje informatike zelo hitro razvija. Napredki so iz dneva v dan večji. S hitrim razvojem tehnologije se pojavlja tudi večja težnja po menjavi zastarele programske in strojne opreme. Na tem področju skoraj ne moremo biti v koraku s časom. Večina, ki se ukvarja z informatiko, se tega zaveda. Razvoj na tem področju je pomemben predvsem zaradi hitrejše odzivnosti in zmogljivosti strojne opreme ter pohitritve procesov, ki jih opravlja računalnik. Prav tako se pozna hiter razvoj na področju programske opreme, saj omogoča skrajšanje postopkov, prijaznejšo uporabo za uporabnika, lažje razumevanje programov itd.

Tematika managementa kakovosti je zelo obsežna in jo je težko definirati v nekaj povedih. To sem spoznala tudi s preučevanjem literature, v kateri sem naletela na različna stališča, mnenja, razumevanje, pojmovanje in opredelitve s strani strokovnjakov tega področja. Večina se jih strinja, da management kakovosti sestavljajo planiranje, zagotavljanje in kontrola kakovosti. Posamezne navedene sestavne dele nekateri tudi nekoliko drugače definirajo. Tak primer je recimo zagotavljanje kakovosti, ki ga mnogi strokovnjaki imenujejo tudi proces izboljšav. Ne glede na to so vsi mnenja, da je kakovost nujno potrebno najprej definirati in jo nato planirati, izvajati, izboljševati in tudi kontrolirati. Prav tako večina avtorjev, ki sem jih v svoje magistrsko delo vključila, meni, da je najpomembnejši proces kontrole, v katerem preverimo, ali produkt dosega ustrezno in zastavljeno kakovost.

Za uspešno izvajanje managementa kakovosti je potrebno dosledno, natančno in stalno delo na področju kakovosti s strani programerjev in testerjev, ki s testiranjem preverjajo ustreznost programske opreme. Ti dve skupini izvajalcev sami ne moreta zagotoviti ustrezne kakovosti, zato jima mora pri tem nujno pomagati tudi naročnik, ki izpostavlja svoje želje, zahteve in pričakovanja končnega izdelka. Naloga projektnega managerja je usklajevanje med naročnikom in svojo projektno skupino, da doseže zastavljene cilje.

Zaradi kompleksnosti kakovosti je potrebno le-to planirati in jo razporediti ter vključiti v vse dele projekta. Preverjanje kakovosti v fazi zaključevanja projekta je prepozno, saj so lahko posledice katastrofalne. Eden izmed razlogov je zagotovo ponovno delo, ki lahko povzroči enormne stroške in dodatno porabi čas za odpravo napak. Noben projektni manager si ne želi prekoračitve planiranih stroškov in časa. Poleg tega je zaradi zaostankov, dodatnih stroškov ter neustrezne funkcionalnosti produkta nezadovoljen tudi naročnik.

Kakovost programske opreme preverjamo s testiranjem, ki ga je potrebno izvajati v vseh fazah in na vseh nivojih projekta. Testiranje opravljata večinoma tester in programer, ki razvija programsko opremo. Če se proces kontrole izvaja stalno, je ob koncu procesa veliko manj napak, težav in nezadovoljstva zaradi neustreznega delovanja programske opreme. Velika večina podjetij, ki se ukvarja razvojem, še vedno ne daje ustreznega poudarka kakovosti skozi celoten življenjski cikel projekta. Posledice so lahko tako drastične, da naročnik končni produkt zavrne. Zavedati se moramo, da na svetu ni programske opreme, ki ne bi imela nobenega hrošča. Slednjega ne moremo zagotoviti že zaradi tega, ker je programska oprema kompleksno zastavljena ter zato, ker programerji ne morejo predvideti vseh možnih uporab razvite programske opreme. Normalno je, da se določeni hrošči pojavijo šele z njeno stalno oziroma z množično uporabo. Naloga podjetij, ki programsko opremo razvijajo, je zagotoviti opremo s čim manj hrošči in nuditi ustrezno odpravljanje napak.

Današnjim podjetjem povzroča veliko težav tudi to, da se ne ločuje med zagotavljanjem in kontrolo kakovosti. Proces zagotavljanja kakovosti vključuje skrb za kakovost in njeno izvajanje, posodabljanje dokumentacije, vpeljavo izboljšav itd. Proces kontrole kakovosti preverja in spremlja kakovost skozi življenjski cikel projekta. Če teh dveh procesov ne ločimo, imamo tudi težave z doseganjem želene kakovosti, saj lahko izpustimo dele, ki so za končni rezultat zelo pomembni. V praksi se največkrat zgodi, da izpuščamo zagotavljanje kakovosti in posledično tudi vpeljave izboljšav. Če želimo to pomanjkljivost odpraviti, je potrebno zaposlene o tem področju ustrezno izobraziti.

Podjetja se skušajo potencialnemu naročniku prikazati v čim boljši luči, zato tudi navajajo svoje pretekle referenčne projekte. V današnjem času se zato podjetja trudijo pridobivati certifikate kakovosti, kot so ISO in IEEE, s katerimi pridobijo na ugledu in povečajo moč svoje blagovne znamke. Certificirana podjetja morajo zagotavljati izvajanje predpisanih postopkov in kontrole ter analizo in spremljanje tekočih podatkov, ki jih posamezni standardi zahtevajo.

Standardi kakovosti so zelo pomemben člen tudi pri oddajanju ponudb naročnikom. Od tega je lahko s strani naročnika odvisen tudi izbor izvajalca. Izpolnjevanje standardov kakovosti nudi podjetjem številne koristi pri trženju njihovih izdelkov ter predstavlja tudi sposobnost doseganja najvišjih standardov za izdelke, ki jih razvijejo, izdelajo.

Za dosego ustrezne kakovosti potrebujemo veliko časa, kadra, natančnosti in vztrajnosti. Upoštevati moramo vse tri procese, ki jih management kakovosti narekuje, ter uporabiti za posamezne primere

ustrezno tehniko, da zagotovimo kar najboljši rezultat. Produktivnost posameznih članov projektne skupine lahko povečamo s tedenskimi sestanki o napredku na projektu, stimulacijo najuspešnejših članov skupine, sproščeno klimo in pravilnim (nenapadalnim) pristopom k članom skupine, ki s svojim delom zaostajajo.

Projektna skupina razvojnega projekta mora delovati vzajemno in stalno iskati izboljšave, ki bi jih lahko vključili v razvoj programske opreme. Poleg navedenega je zelo pomembno tudi stalno spremljanje in kontroliranje kakovosti skozi življenjski cikel projekta, saj omogoča pravočasno odkrivanje napak in njihovo odpravo.

LITERATURA IN VIRI

1. *About IEEE* (b.l.). Najdeno 10. novembra 2011 na spletnem naslovu http://www.ieee.org/about/index.html
2. Ammann, P., & Offutt, J. (2008). *Introduction to Software Testing*. New York: Cambridge University Press.
3. Baresi, L., & Pezzè, M. (2006). An Introduction to Software Testing. *Electronic Notes in Theoretical Computer Science,* str. 89–111.
4. *Basic Tools for Process Improvement – Module 6: Flowchart*. Najdeno 22. novembra 2011 na spletnem naslovu http://www.doh.state.fl.us/hpi/pdf/Flowchart.pdf
5. *Basic Tools for Process Improvement – Module 10: Control Chart*. Najdeno 22. novembra 2011 na spletnem naslovu http://www.doh.state.fl.us/hpi/pdf/ControlChart2.pdf
6. *Basic Tools for Process Improvement – Module 5: Cause-and-Effect Diagram*. Najdeno 22. novembra 2011 na spletnem naslovu http://www.doh.state.fl.us/hpi/pdf/Cause%26EffectDiagram2.pdf
7. *Basic Tools for Process Improvement – Module 9: Run Chart*. Najdeno 20. novembra 2011 na spletnem naslovu: http://www.au.af.mil/au/awc/awcgate/navy/bpi_manual/mod9-runchart.pdf
8. Bell, D. (2003). UML basics: An introduction to the Undified Modeling Language. Najdeno 16. avgusta 2012 na spletnem naslovu http://www.nyu.edu/classes/jcf/g22.2440-001_sp06/handouts/UMLBasics.pdf
9. Bøegh, J., Depanfilis, S., Kitchenham, B., & Pasquini, A. (1999). A Method for Software Quality Planning, Control, and Evaluation. *IEEE Software: Researcher's Corner,* str. 69–78
10. Botrlla, P., Burgués, X., Carvallo, J. P., Franch, X., Grau, G., Marco, J., & Quer, C. (b.l). ISO/IEC 9126 in practice: what do we need to know?. Najdeno 28. oktobra 2011 na spletnem naslovu http://www.essi.upc.edu/~webgessi/publicacions/SMEF'04-ISO-QualityModels.pdf
11. Božič, M. (2004). *Novosti in učinki uvajanja ISO standardov*. Ljubljana: Ekonomska fakulteta.
12. Burton Swanson, E. (1997). Maintaining IS quality, The Aderson School at UCLA, Los Angeles, USA. *Information and software Technology,* 39, 845–850.
13. Cai, K.-Y., Li, Y.-C., & Ning, W.-Y. (2005). Optimal software testing in the setting of controlled Markov chains. *European Journal of Operational Research,* 162, 552–579.
14. *Cause&Effect Diagram* (b.l.). Najdeno 22. novembra 2011 na spletnem naslovu http://www.skymark.com/resources/tools/cause.asp
15. Champman, P. (2012). SEC Approves NYSE Retail Liquidity Program. Najdeno 8. avgusta 2012 na spletnem naslovu http://www.tradersmagazine.com/news/retail-110114-1.html?pg=1
16. *CIMO guide* (2008). 7th ed. Najdeno 28. oktobra 2011 na spletnem naslovu http://www.wmo.int/pages/prog/www/IMOP/publications/CIMO-Guide/CIMO%20Guide%207th%20Edition,%202008/Part%20III/Chapter%201.pdf
17. *Control Charts* (b.l.). Najdeno 26. oktobra 2011 na spletnem naslovu http://www.statlets.com/control_charts.htm
18. *Cost of Quality* (2008, 12. september). Najdeno 2. novembra 2011 na spletnem naslovu http://totalqualitymanagement.wordpress.com/category/cost-of-quality/
19. Čibej, M. (2010, 22. november). Zakaj IT projekti ne uspejo?. *Glasilo Združenja za projektni management PMI Slovenija*. Najdeno 28. oktobra 2011 na spletnem naslovu http://www.pmi-

slo.org/PMI/PMI.nsf/0/b1df530dc4897165c12576ea002c221e/$FILE/PMI_SLO%20Newsletter%202010_03.pdf
20. Dalal, S., Jain, A., Karunanithi, N., Leaton, J. M., Lott, C. M., Patton, G. C., & Horowitz, B. M. (1999). Model-based testing in practice. *Proceedings of ICSE'99: ACM Press,* 1–10.
21. *Data Collection and Analysis Tools*. Najdeno 10. decembra 2011 na spletnem naslovu: http://asq.org/learn-about-quality/data-collection-analysis-tools/overview/overview.html
22. Dias-Neto, A., C., & Travassos, G., H.(2009). Model-based testing approaches selection for software projects. Federal University of Rio de Janeiro. *Information and Software Technology* , 51, 1487– 1504.
23. Elberzhager, F., Münch, J., & N. Nha, V., T. (2011). A systematic mapping study on the combination of static and dynamic quality assurance techniques, *Information and Software Technology*, 54, 1 – 15.
24. *Six Sigma Traning Resources* (b.l.). Najdeno 10. novembra 2011 na spletnem naslovu http://www.dmaictools.com/excel-templates
25. Field, M., & Keller, L. (2007). *Project Management*. London:The Open University
26. *Flow Chart* (b.l.). Najdeno 22. novembra 2011 na spletnem naslovu http://www.mindtools.com/pages/article/newTMC_97.htm
27. Futrell, R. T., Shafer, D. F., & Shafer, L. I. (2002). *Quality software project management*. New Jersey: Prentice Hall PTR.
28. Galin, D. (2004). *Software quality assurance: from theory to implementation*. Harlow: Pearson Education Limited.
29. Gider, F. (2011). Tehnike reševanja problema. Najdeno 22. novembra 2011 na spletnem naslovu http://robo.fe.uni-lj.si/~imrec/predavanjeMSI/MSI%20-%20uvod%20v%20predmet%202010-10.pdf
30. Goncalves, D., Varejao, J., Martinho, R., & Cruz, J., B. (2010). *Quality Management: Concepts and Approaches for software Projects*. New York: Springer.
31. *History of IEEE* (b.l.). Najdeno 10. novembra 2011 na spletnem naslovu http://www.ieee.org/about/ieee_history.html
32. Ho, S., & Fung, C. (2008). TQMEX Model. Najdeno 22. novembra 2011 na spletnem http://www.hk5sa.com/tqm/tqmex/intro.htm
33. *IEEE Standard Association* (b.l.). Najdeno 10. novembra 2011 na spletnem naslovu http://odysseus.ieee.org/query.html?qt=&charset=iso-8859-1&style=standard&col=sa
34. *IEEE Standard association: 1012 – 2004* (b.l.). Najdeno 10. novembra 2011 na spletnem naslovu http://standards.ieee.org/findstds/standard/1012-2004.html
35. *IEEE Standard Association: 1008 – 1987* (b.l.). Najdeno 10. novembra 2011 na spletnem naslovu http://standards.ieee.org/findstds/standard/1008-1987.html
36. IPCC (2000). IPCC Good Practice Guidance and Ucertainly Management in National Greenhouse Gas Inventories. Najdeno 10. decembra 2011 na spletnem naslovu http://www.ipcc-nggip.iges.or.jp/public/gp/english/index.html
37. *ISO* (b.l.). Najdeno 10. decembra 2011 na spletnem naslovu http://www.iso.org/iso/qmp_2012.pdf

38. *ISO 9000* (b.l.). Najdeno 10. decembra 2011 na spletnem naslovu http://www.iso.org/iso/home/standards/management-standards/iso_9000.htm
39. *ISO 9126* (b.l.). Najdeno 10. decembra 2011 na spletnem naslovu http://www.angelfire.com/nt2/softwarequality/ISO9126.pdf
40. *SQA* (b.l.). ISO 9126 - Software Quality Characteristics. Najdeno 26. oktobra 2011 na spletnem naslovu http://www.sqa.net/iso9126.html
41. Jarrett, J. (1995). Review of a Quality Control and Improvement statistical Software System. University of Rhode Island, Kingston, ZDA. The statistical software newsletter, 333–339.
42. Kakovost (b.l.). V *SSKJ*. Najdeno 12. novembra 2011 na spletnem naslovu http://bos.zrc-sazu.si/cgi/a03.exe?name=sskj_testa& expression=kako vost &hs=1
43. Kerner, C. (2003, maj). What Is a Good Test Case?. Florida Institute of Technology. Najdeno 9. decembra 2011na spletnem naslovu http://www.kaner.com/pdfs/GoodTest.pdf
44. Kloppenborg, T., J., Patrick, J., & Petrick, A. (2002). Managment project quality, Management concepts, USA. *Information and Software Technology*, 54, 1–15
45. Kobayashi, N., Tsuchiya, T., & Kikuno, T. (2002). Non-specification-based approaches to logic testing for software. *Information and Software Technology*, 44,113–121.
46. Kousholt, B. (2007). *Project Management: Theory and practice* (1st ed.). Copenhagen: Nyt Teknisk Forlag.
47. Marchewka, J., T. (2002). *Information technology project management: Providing measurable organisational value* (1st ed.). New Jersey: Whiley.
48. Mathur, A., P. (2011). *Foundation of Software Testing*. New Delhi: Pearson Education.
49. Naik, K., & Tripathy, P. (2008). *Software Testing and Quality Assurance: Theory and Practice*. New Jersey: Wiley.
50. Olds, D., & Consulting, G. (2012). How one bad algorithm cost traders $440m. Najdeno 10. avgusta 2012 na spletnem naslovu http://www.theregister.co.uk/2012/08/03/bad_algorithm_lost_440_million_dollars/
51. *Pareto chart* (b.l.). Najdeno 20. novembra 2011 na spletnem naslovu http://personnel.ky.gov/NR/rdonlyres/D04B5458-97EB-4A02-BDE1-99FC31490151/0/ParetoChart.pdf
52. *Pareto Charts* (b.l.). Najdeno 20. novembra 2011 na spletnem naslovu http://www.qualityamerica.com/knowledgecente/knowctrPARETO_CHARTS.htm
53. *Pareto*. Najdeno 10. decembra 2011 na spletnem naslovu http://asq.org/learn-about-quality/cause-analysis-tools/overview/pareto.html
54. Patton, R. (2005). *Software Testing* (2nd ed.). ZDA: SAMS Puhlishing.
55. *Perform Quality Control* (b.l.). Najdeno 30. oktobra 2011 na spletnem naslovu http://www.mypmps.net/en/mypmps/knowledgeareas/quality/perform-quality-control.html
56. Pezzè, M., & Young, M. (2007). *Software Testing and analysis: Process, Principles and Techniques*. New Jersey: Wiley.
57. PMI (2008). *PMBOK Guide*. Newton Square: PMI.
58. *Poslovna strategija SIX SIGMA* kaj je to (b.l.). Najdeno 30. oktobra 2011 na spletnem naslovu http://www.ibsporocevalec.si/component/content/article/36-marec/86-poslovna-strategija-six-sigma-kaj-je-to-

59. Potkonjak, P. (b.l.). *Spletni priročnik: Kakovost poslovanja.* Najdeno 12. novembra 2011 na spletnem naslovu http://www.mojdenar.com/alea/dokumenti/dokument.asp?id=14#kaz2
60. *Project Quality Planning* (2004, september). Najdeno 28. oktobra 2011 na spletnem naslovu http://www.projectperfect.com.au/info_project_quality_planning.php
61. *Quality Planning* (b.l.). Najdeno 30. oktobra 2011 na spletnem naslovu http://www.mypmps.net/en/mypmps/knowledgeareas/quality/quality-planning.html
62. Reeves, C., A., & Bednar, D., A. (1994). Defining quality: alternatives and implications. *Academy of Management Review,* 19, 419–445.
63. Ricca, F., Torchiano, M., Di Penta, M., Ceccato, M., & Tonella, P. (2009). Using acceptance tests as a support for clarifying requirements: A series of experiments. *Information and Software Technology,* 51, 270–283.
64. Rose Kenneth, H. (2005). *Project Quality Management: Why, What and How*. Boca Raton: J.Ross publishing.
65. Sampath, S., Sprenkle, S., Gibson, E., Pollock, L., & Greenwald, A., S. (2007). Applying Concept Analysis to User-Session-Based Testing of Web Applications. *IEE Transactions on Software Engineering,* 643–657.
66. *Scatter Diagram* (b.l.). Najdeno 30. oktobra 2011 na spletnem naslovu http://web2.concordia.ca/Quality/tools/25scatter.pdf
67. *Scatter Diagrams* (b.l.). Najdeno 30. oktobra 2011 na spletnem naslovu http://personnel.ky.gov/NR/rdonlyres/CF0C40D5-7E53-4DEE-9FC6-EF0FBBEEBD30/0/ScatterDiagrams.pdf
68. Schwalbe, K. (2007). *Information Technology Project Management* (5th ed.). Boston: Course Technology.
69. Schwalbe, K. (2009). *Information Technology Project Management* (6th ed.). Boston: Course Technology.
70. Sidoroska Ćorić, D. (2005, december). *Benchmarking v Adrii Airways* (Diplomsko delo). Ljubljana: Ekonomska fakulteta.
71. Siek, K.(2002). Run Chart, Boise state university. Najdeno 30. oktobra 2011 na spletnem naslovu http://www.freequality.org/documents/knowledge/RunChart.pdf
72. Silver-Greenberg, J., Popper, N., & De La Merced, M., J. (2012). Trading Program Ran Amok, With No »Off« Switch. Najdeno 10. avgusta 2012 na spletnem naslovu http://dealbook.nytimes.com/2012/08/03/trading-program-ran-amok-with-no-off-switch/
73. SIST (b.l.). Sistem vodenja kakovosti in slovenski standard. Najdeno 29. oktobra 2011 na spletnem naslovu http://www.sist.si/index.php?option=com_content&view=article&id=112&catid=39&Itemid=161
74. Sommerville, I. (2004). Software Engineering, Lecture on Quality Managementu. Najdeno 28. oktobra 2011 na spletnem naslovu http://www.comp.lancs.ac.uk/computing/resources/IanS/SE7/Presentations/PDF/ch27.pdf
75. *Strategic Quality Planning* (2008, 4. oktober). Najdeno 30. oktobra 2011 na spletnem naslovu http://totalqualitymanagement.wordpress.com/2008/10/04/strategic-quality-planning/

76. Sumramanian, G., H., J. Jiang, J., & Klein, G. (2007). Software quality and IS project performance improvements from software development process maturity and IS implementation strategies. *Journal of Systems and Software,* 80, 616–627.
77. Šircelj, J. (2006, september). *Standard kakovosti ISO 14001 kot vir konkurenčne prednosti: Primer podjetja TIB transport d.d.* (diplomsko delo). Ilirska Bistrica: Ekonomska fakulteta.
78. Štempihar, A. (2010, 22. november). Poslovni analitik v IT projektih, 2010. *Glasilo Združenja za projektni management PMI Slovenija*. Najdeno 28. oktobra 2011 na spletnem naslovu http://www.pmi-slo.org/PMI/PMI.nsf/0/b1df530dc4897165c12576ea002c221e/$FILE/PMI_SLO%20Newsletter%202010_03.pdf
79. Takahashi, T. (1996). Statistical Games and Software Tools for Quality Assurance Based on Statistical Process Control; Science University of Tokyo, Japan. *Computers ind. Engng,* 31, 759 – 765.
80. *Tehnike reševnja problemov*. Najdeno 20. novembra 2011 na spletnem naslovu www.abakus.si/wik**i_ima**ges/.../5/53/Tehnike_resevanja_problemov.ppt
81. Tepandi, J. (1997). Quality Assurence of Knowledge-Based Systems. *Elsevier Science Ltd,* 10, 231 – 241.
82. *The Histogram* (b.l.). Najdeno 20. novembra 2011 na spletnem naslovu http://www.netmba.com/statistics/histogram/
83. Toplak B., M., & Urbajs, A. (2003). Kakovost po ISO 9001:2000. *Institut informacijskih znanosti(IZUM)*. Najdeno 26. oktobra 2011 na spletnem naslovu http://home.izum.si/COBISS/OZ/2003_3/html/clanek_02.html
84. *Uspešna vpeljava*. Najdeno 30. oktobra 2011 na spletnem naslovu http://inform.6-sigma.info/uspesnavpeljava
85. Wallance, S. (2004). The ePMbook – Project and Programme Management. Najdeno 30. oktobra 2011 na spletnem naslovu http://www.epmbook.com/
86. What is ISO 9004 (b.l.). Najdeno 11. novembra 2011 na spletnem naslovu http://the9000store.com/what-is-iso-9004.aspx
87. Williams, C. (2011, 11. september). The $300m cable that will save traders milliseconds. Najdeno 10. avgusta 2012 na spletnem naslovu http://www.telegraph.co.uk/technology/news/8753784/The-300m-cable-that-will-save-traders-milliseconds.html
88. Zakon o standardizaciji. *Uradni list RS* št. 1/1995.
89. Zhao, L., & Elbaum, S. (2003). Quality assurance under the open source development model. *The Journal of Systems and Software,* 66, 65–75.