UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

DIPLOMSKO DELO

UVAJANJE STANDARDA ISO 9001 V PODJETJE Z AGILNIM RAZVOJEM PROGRAMSKE OPREME

V Ljubljani, april 2011 IVO ZGONIK

IZJAVA

Študent Ivo Zgonik izjavljam, da sem avtor/ica tega diplomskega dela, ki sem ga napisal/a pod mentorstvom prof. dr. Boruta Rusjana, in da v skladu s 1. odstavkom 21. člena Zakona o avtorskih in sorodnih pravicah dovolim njegovo objavo na fakultetnih spletnih straneh.

V Ljubljani, dne_____ Podpis: _____

KAZALO

KAZALO SLIK

UVOD

Kot vsako podjetje se tudi podjetja, ki razvijajo programsko opremo, trudijo kar najbolj znižati stroške, povečati zanesljivost izdelka, zmanjšati čas razvoja in, kar je najpomembneje, zagotoviti, da bo izdelek tak, kakršnega si želi kupec.

Tradicionalne metode razvoja programske opreme so se začele na začetku sedemdesetih let in jih je strokovna javnost kot prve uspešne procesno določene metode dela pri razvoju programske opreme hitro zelo dobro sprejela. Bile so nujo potrebne za velike projekte, pri katerih so bile znane natančne lastnosti bodočega izdelka in so tako omogočale točen popis zahtev, ki jih mora izdelek izpolnjevati. V realnosti, še posebej v zadnjih letih, je takšnih izdelkov malo. Zelo pogosto se naročnik v času razvoja izdelka premisli o določenih značilnostih ali pa med projektom celo ugotovi, da so določene zahteve nerealne, oziroma si nasprotujejo. Ker lahko projekti razvoja določene programske rešitve trajajo tudi več let, se v tem času lahko spremeni zakonodaja in postanejo določene vnaprej predvidene zahteve nepotrebne oziroma nesmiselne. Razvoj izdelka, katerega prva faza zahteva natančen načrt celotnega izdelka, da popiše vse lastnosti izdelka in do izvedbe le-tega pride šele v drugi fazi projekta, lahko postane zelo neučinkovit.

Zaradi takšnih sprememb večina projektov razvoja programske opreme ni uspešnih glede na prvotne načrte. Zato so se v devetdesetih letih prejšnjega stoletja začele razvijati nove metodologije razvoja programske opreme, tako imenovane agilne metodologije razvoja, s katerimi je laže doseči lastnosti iz prvega odstavka. Omogočajo veliko večjo prožnost, končni uporabnik oziroma naročnik izdelka lahko oziroma mora veliko bolj sodelovati pri razvoju izdelka, zagotavljanje in preverjanje kakovosti pa večinoma poteka sproti in ne le po končni fazi razvoja.

Kljub temu, da agilne metodologije v veliko večji meri zagotavljajo, da bo razvoj izdelka težil k lastnostim, ki jih v resnici želi končni uporabnik, se je marsikatero podjetje srečalo s povsem drugačnim problemom. Ta je jedro pričujoče diplomske naloge. Bistvo agilnih metodologij razvoja programske opreme se, vsaj na videz, kreše z ISO standardom kakovosti (ISO 9001), ki so zelo procesno naravnani in pogosto, vsaj navidezno, zahtevajo veliko dokumentacije. ISO standard kakovosti ISO 9001:2000 so namreč nastali še v času, ko se je o agilnih metodah razvoja programske opreme šele razpravljalo, zato temeljijo na tradicionalnih metodah razvoja programske opreme, predvsem na Waterfall oziroma t.i. slapovni metodologiji. Kasnejši ISO 9001:2008 ni bistveno spremenil zahtev za skladnost. Ena od zahtev agilnih metod razvoja, da se dela le tisto, kar je nujno potrebno za sam razvoj (Beck, 2001) in se zato ne sme pripravljati nobene odvečne dokumentacije, hitro pripelje do sklepa, da je izpolnjevanje ISO standardov v takšnem podjetju lahko zelo obremenilno za uspešen in učinkovit proces. Pogosto razumevanje ISO standarda kakovosti je tudi, da morajo biti vse značilnosti bodočega izdelka znane vnaprej, ob podpisu pogodbe. Ta zahteva je v nasprotju z agilnimi metodami razvoja, ki zahtevajo, da so vnaprej znane le t.i. uporabniške zgodbe, in je

največji izziv pri uvajanju ISO standarda v podjetje z agilnimi metodami razvoja (Stålhane & Hanssen, 2008).

Kombiniranje ISO standarda kakovosti z agilnim razvojem programske opreme se med avtorji, ki raziskujejo to tematiko, zelo razlikuje. Nekateri pravijo, da v danem primeru skladnost z ISO standardi ni potrebna, saj že same agilne metode zagotavljajo primerno raven kakovosti. S takšnim posploševanjem se večina avtorjev ne strinja, ni pa tudi enotnega mnenja, koliko trpijo osnovne vrednote agilnih metod pri uvajanju standardov.

Poudaril bi pomen ISO standardov, ker so splošno sprejeti standard, da lahko tako notranji kot zunanji udeleženci določenega procesa verjamejo, da podjetje procesu res sledi. Pomembno je, da zunanji revizor pregleda procese in potrdi, da v njih ne obstajajo luknje, ki bi lahko ogrozile izvedbo ciljev.

Moj namen pri pisanju diplomske naloge je povzeti rešitve iz strokovne in znanstvene literature, ki opisujejo uvajanje ISO standarda kakovosti v podjetje z agilnimi metodologijami razvoja programske opreme, na čim uporabnejši način za podjetja, ki naletijo na ta izziv v praksi.

Cilj diplomske naloge je predstaviti način, kako agilne metodologije razvoja programske opreme zagotavljajo kakovost in kako poteka zagotavljanje kakovosti skladno z ISO standardom kakovosti, kje se metodologije razvoja skladajo s standardom in kje se lahko pojavijo ovire pri usklajevanju.

Diplomsko delo je sestavljeno iz štirih delov:

- 1. del diplomskega dela definira kakovost in predstavi problematiko nekakovostnih izdelkov v IT industriji.
- 2. del predstavlja metodologije razvoja programske opreme. Okvirno sta predstavljeni tradicionalni metodologiji slapovni razvoj in t.i. V-proces, bolj podrobno pa agilne metodologije, kjer se osredotočam na najpogosteje uporabljeni metodologiji Scrum in XP. Pri opisu različnih metodologij razvoja programske opreme se bom osredotočil na področje zagotavljanja kakovosti.
- 3. del predstavlja standard kakovosti ISO 9001, njegova načela, prednosti in slabosti ter vsebino. V tem poglavju navajam tudi druge standarde ISO, ki jih je treba razumeti pri uvajanju ISO 9001.
- 4. del poveže standard ISO 9001 in agilne metodologije razvoja programske opreme. Najprej so predstavljene raziskave drugih avtorjev o tej tematiki, sledi pregled standarda ISO 9001 po točkah in z ustreznim opisom, kako določeno zahtevo izpolniti v podjetju, ki uporablja agilne metodologije razvoja in kje lahko pričakujemo večje izzive. Na koncu poglavja navajam prilagoditve načina dela za podjetje, ki uporablja katero od agilnih metodologij, ki jih imam za najpomembnejše.

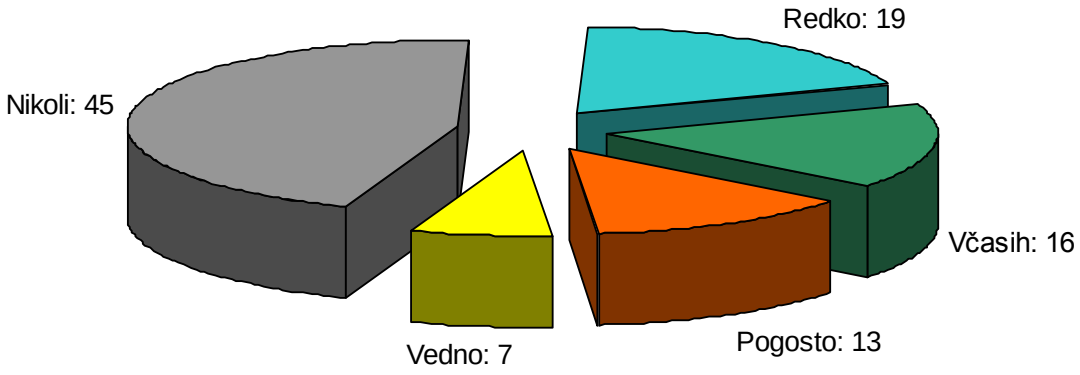
1 KAKOVOST PROGRAMSKE OPREME

Definicij kakovosti je verjetno toliko kot avtorjev, ki pišejo o njej. Večkrat citirana in v praksi uporabljana je definicija, ki je opredeljena v standardu ISO 9000: Sistemi vodenja kakovosti – Osnove in slovar: »Kakovost je stopnja, v kateri skupek svojstvenih karakteristik izpolnjuje zahteve« (Slovenski inštitut za standardizacijo, 2005).

Razmeroma drugače je definirana kakovost pri agilnih metodologijah razvoja programske opreme. Janet Gregory, soavtorica knjige Agile Testing (Crispin & Gregory, 2009), v svojem članku (Gregory, 2010) definira, da je kakovosten tisti izdelek, ki doseže oziroma preseže naročnikova oziroma kupčeva pričakovanja. Sama sicer prizna, da ta definicija pogosto ne zadostuje, vendar poudarja, da si je, ko izdelek doseže minimalne standarde, treba prizadevati za doseganje in preseganje naročnikovih pričakovanj.

Če primerjamo več izdelkov, je pogosto zelo težko določiti, kateri je bolj kakovosten, vedno pa je rezultat stvar subjektivne presoje, ne glede na samo definicijo kakovosti. Laže kot kakovost je meriti uporabnost izdelka, ki je pogosto zajeta v različnih definicijah kakovosti.

Slika 1: Uporabljanost funkcionalnosti v povprečnem programu (v %).



Vir: Povzeto po J. Johnson, Build only the Features You Need, 2002.

Da izdelki programske opreme ne blestijo po uporabnosti, nazorno kaže Slika 1, saj po raziskavi Standish Group kar 45 % funkcionalnosti v povprečnem programu ni nikoli uporabljenih. Redko ali včasih je uporabljanih 35 %, pogosto ali vedno uporabniki uporabljajo le petino funkcionalnosti (Johnson, 2002).

Podatki nazorno kažejo, da je eden večjih izzivov pri razvoju programske opreme delati manj nepotrebnih stvari.

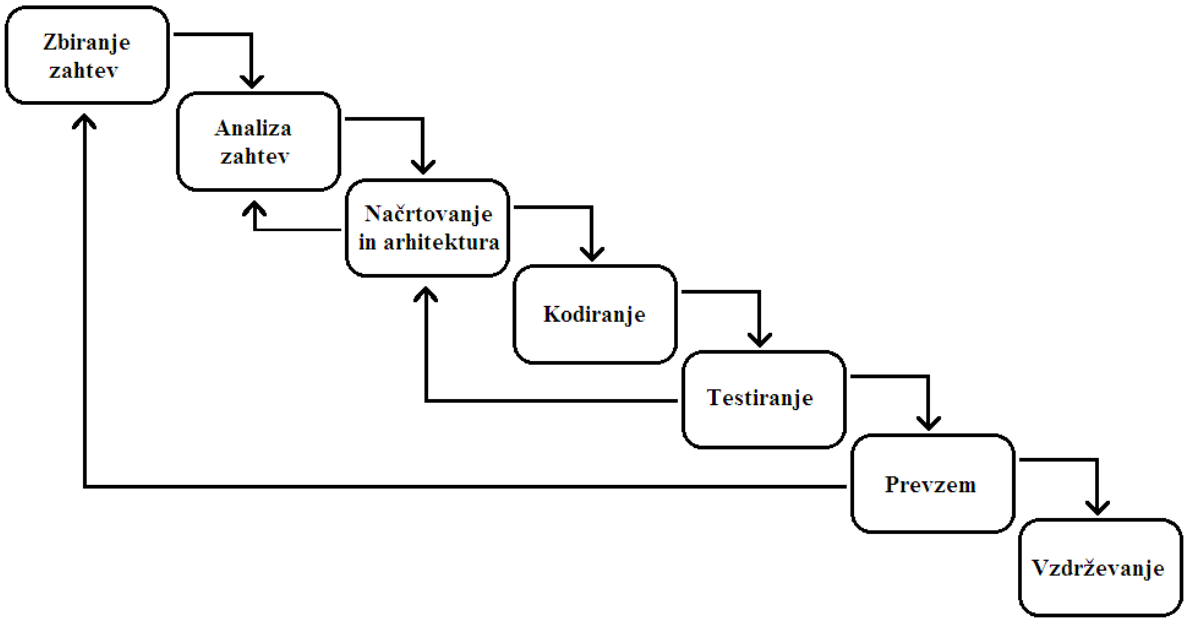
2 METODOLOGIJE RAZVIJANJA PROGRAMSKE OPREME

2.1 Klasične metodologije razvoja programske opreme

2.1.1 Slapovni model

Slapovni model (angl. *Waterfall model*) razvoja programske opreme je že leta 1970 predstavil Winston W. Royce. Je najstarejši, največkrat citiran in uporabljan model. Slapovni model razvoja je izrazito procesno naravnan, saj si faze sledijo ena za drugo in ena se lahko začne šele, ko se predhodna konča. Podjetja pogosto uporabljajo različne oddelke za vsako fazo razvoja ali pa se določena faza izvede celo v drugem podjetju (Waterfall model, 2011).

Slika 2: Slapovni model razvoja programske opreme



Vir: povzeto po S. Shore & S. Warden, The art of Agile Developement, 2007.

Različni viri razčlenijo slapovni model razvoja programske opreme na različne faze, za vse pa je značilno, da potekajo zaporedno in da se napako, odkrito v eni fazi, odpravlja v predhodnih fazah. Na Sliki 2 je prikazana najbolj pogosta razčlenitev slapovnega modela: puščice navzdol prikazujejo redno sledenje faz, puščice navzgor pa vračanje na predhodne faze, če se odkrije nepravilnost. Spodaj opisane faze lahko trajajo različno dolgo, odvisno od posameznega projekta (Shore & Warden, 2007).

- **Zbiranje in analiza zahtev** – Prva faza obsega zbiranje naročnikovih zahtev in podrobno proučitev, katere zahteve bi lahko bile nerealne ali nesmiselne, saj si lahko med seboj nasprotujejo. Ko so zahteve jasno zapisane, jih naročnik potrdi, da se po njih napiše specifikacije, namenjene tako naročniku kot razvijalcem. Ta dokumentacija je jasna in razumljiva tako naročniku kot izvajalcem.

- **Načrtovanje in arhitektura** – Glavna značilnost tradicionalnih metodologij razvoja programske opreme se pokaže v tej fazi s t.i. »Big Design Up Front«, kar pomeni, da še pred začetkom izvedbe naredijo natančen načrt izdelka.

  Ta faza ima nekaj izrazitih prednosti in slabosti. Včasih se izkaže, da prvotni načrt ni izvedljiv. V takšnem primeru je sprememba načrta veliko cenejša v tej, začetni fazi, kakor če bi bil ta problem odkrili šele med implementacijo (kodiranjem) posameznih delov produkta. Po drugi strani takšen proces dela onemogoča oziroma zelo oteži kakršno koli spremembo naročnikovih želja po zaključku te faze. Velikokrat se med implementacijo tudi izkaže, da je bila kljub podrobnemu načrtovanju v začetku projekta kaka pomembna podrobnost spregledana, in se je iz faze izvedbe treba vrniti nazaj v fazo načrtovanja, kar lahko zelo podaljša čas izvedbe produkta.

- **Izvedba načrtovanega** – Faza kodiranja oziroma preslikave načrta v programsko kodo da najbolj vidne rezultate v procesu razvoja, čeprav ta faza ni nujno najobsežnejša. Posamezne dele programskega paketa se integrira v celoto proti koncu te faze.

- **Testiranje** – Faza testiranja obsega preverjanje, ali programski paket deluje po specifikaciji. Opraviti Izvesti je treba vsebinske teste, testiranje robustnosti in testiranje učinkovitosti programskega paketa. Vsebinski testi potrdijo, da so narejene vse specificirane funkcionalnosti. Testiranje robustnosti potrdi, da programskega paketa ni mogoče pripraviti do nepredvidenega oziroma neželenega stanja.

Nasprotniki slapovnega procesa običajno najbolj kritizirajo prav to fazo, saj se tveganje projekta poveča, če je testiranje šele na koncu. Naročnik do zadnjega trenutka ne more biti prepričan, ali bo dobil naročeno pravočasno ali ne. V fazi testiranja lahko pride na dan veliko napak, ki jih je treba odpraviti, kar podaljša trajanje in poveča ceno projekta. Po končani fazi testiranja oziroma, ko testnimi primeri dokažejo pravilno delovanje programskega paketa, se-le tega inštalira pri naročniku. Trajanje faze inštalacije je odvisno od posameznega projekta.

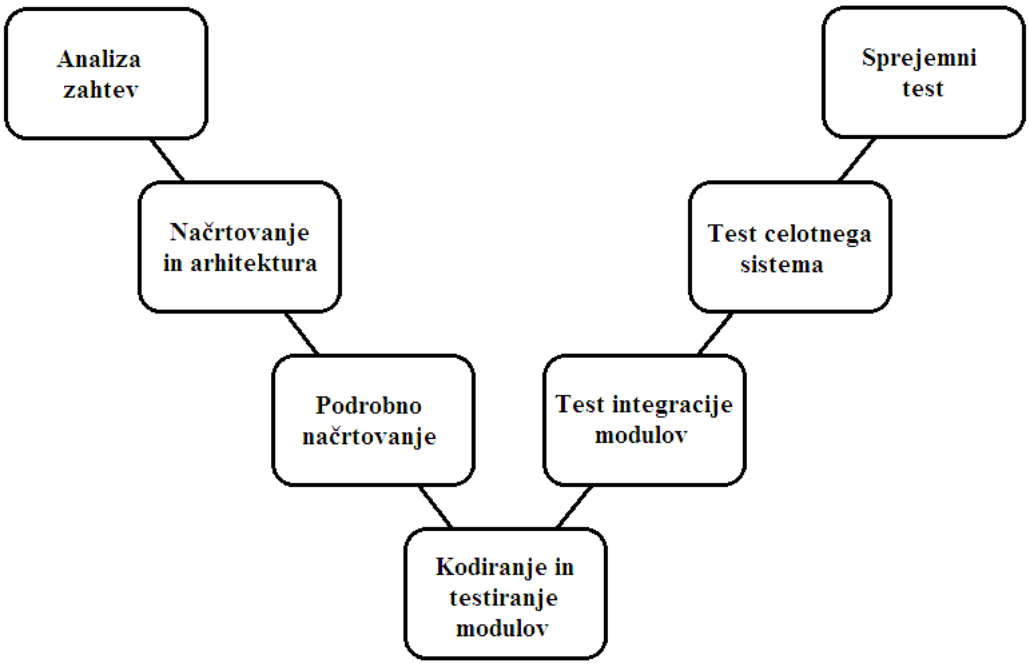
- **Prevzem** – To fazo opravi naročnik, potem ko je aplikacijo že prejel. Njegova naloga je preveriti, ali bo programski paket služil svojemu namenu. Tako kot faza testiranja je tudi faza prevzema pogosta tarča kritikov slapovnega procesa, saj lahko naročnik ugotovi, da ni dobil natančno tistega, kar je pričakoval, čeprav je bil projekt izveden dosledno. V takem primeru je treba spremeniti začetne zahteve.

- **Vzdrževanje** – Ta faza pomeni podporo uporabnikom za čim učinkovitejšo uporabo programskega paketa. V tej fazi odpravljajo tudi napake, ki jih je odkril naročnik, izboljšujejo programski paket, da sledi novim tehnologijam, in ga dopolnjujejo z novimi funkcionalnostmi.

2.1.2 V-model

V-model razvoja programske opreme temelji na slapovnem modelu, vendar je učinkovitejši pri pravočasnem zagotavljanju kakovosti. Glavna prednost V-modela je, da se proces testiranja premakne v zgodnejše faze razvoja. Faza testiranja je razširjena in se jo izvaja med fazami: analiza zahtev in priprava specifikacij, načrtovanje arhitekture, podrobno načrtovanje in faza izvedbe. Kot prikazuje organigram v Sliki 3, se za vsako od omenjenih faz pripravi plan, kako se jo bo testiralo. Sam proces testiranja še zmeraj poteka po implementaciji, vendar so testni plani in včasih tudi avtomatski testi že napisani, zato poteka veliko hitreje (V-Model (software development), 2011; Waterfall model, 2011).

Slika 3: V-model razvoja programske opreme



Vir: povzeto po S. Shore & S. Warden, The art of Agile Developement, 2007.

V-model razvoja programske opreme je razširjena različica slapovnega modela, zato si posamezne faze razvoja še zmeraj sledijo zaporedno. Ker pa se faza testiranja začne hkrati s preostalimi fazami razvoja programske opreme, je V-model prvi korak k agilnim metodologijam, ki so opisane v poglavju 2.2. Posamezne faze testiranja pri razvoju programske opreme so (Shore & Warden, 2007):

- Testiranje modulov (angl. *Unit tests*) – Test posameznih enot programskega paketa je najnižja stopnja testiranja in jo navadno izvajajo razvijalci te enote.

- Test integracije modulov (angl. *Integration test*) – Ko je delovanje posameznih enot zagotovljeno, se preveri medsebojno delovanje enot programskega paketa. To fazo procesa testiranja navadno opravlja tester. To je oseba, ki je določena za zagotavljanje kakovosti.

- Test celotnega sistema (angl. *System test*) – test celotnega sistema programskega paketa naj bi zagotovil, da bo programski paket opravil sprejemni test oziroma test naročnikovih zahtev. Test sistema testira tudi nefunkcionalne vidike programskega paketa, se pravi učinkovitost, hitrost in robustnost na velike obremenitve.

- Sprejemni test (angl. *Acceptance test*) je zadnji test pred predajo programskega paketa uporabniku. Pri tem testu ponavadi v večji meri sodeluje naročnik oziroma ga v celoti opravi sam. Ta test preveri, ali programski paket zadošča poslovnim potrebam, ali mu uporabnik lahko zaupa in ali je za uporabnika (dovolj) uporaben.

V-model razvoja programske opreme je izpopolnjena oblika slapovnega modela, predvsem kar zadeva zagotavljanje kakovosti. Kljub temu da ta model veliko bolj formalizira procese zagotavljanja kakovosti, je deležen večinoma enakih kritik kot slapovni model. Te so (Shore & Warden, 2007):

- Naročnik ima možnost videti razvoj produkta šele, ko je ta končan, kar je lahko zelo tvegano. V primeru, da naročnikove zahteve niso ustrezale njegovim dejanskim potrebam, je izgubljen velik del investicije.

- Tako implementacija kot testiranje implementiranega se začneta šele pozno v življenjskem ciklu razvoja programske opreme, kar povečuje tveganje, da izdelek nčan v roku.

2.2 Agilne metodologije razvoja programske opreme

V Združenih državah Amerike se je leta 2001 sestala skupina strokovnjakov s področja razvoja programske opreme, ki je želela združiti najboljše prakse te industrijske veje in z večjim posluhom za ljudi izboljšati rezultate razvoja programske opreme. Nastal je Agilni manifest (angl. *Agile Manifesto*), temeljna listina za agilni razvoj programske opreme. Določa načela in priporočila agilnih metodologij (History: The Agile Manifesto, 2001).

2.2.1 Glavne vrednote agilnih metodologij

Drugače od tradicionalnih metodologij razvoja programske opreme, ki temeljijo na planiranju, temeljijo agilne metodologije na vrednotah, zapisanih v Agilnem manifestu, ki ga prikazuje Slika 4. Posvečajo se vprašanju, kaj je pomembnejše in na katere dejavnike se je treba osredotočiti pri razvoju programske opreme.

Slika 4: Agilni manifest

»Z razvojem programske opreme in s pomočjo drugim pri razvijanju odkrivamo boljše načine za doseganje cilja. Pri tem delu smo prišli do naslednjih vrednot:

Posamezniki in njihovo medsebojno sodelovanje proti procesom in orodjem
Delujoča programska oprema proti popolni dokumentaciji
Sodelovanje naročnika proti pogajanju o pogodbi
Odzivanje na spremembe proti sledenju planom

Postavke na desni sicer imajo svojo vrednost, vendar bi radi poudarili, da vrednotimo postavke na levi više.«

Vir: Manifesto for Agile Software Development, 2001.

Agilne metodologije poudarjajo pomembnost posameznika, sodelovanja med razvijalci in spodbudnega okolja med njimi. Procesi in orodja morajo biti takšni, da olajšajo delo razvijalcem, saj dober proces resda ne more rešiti projekta, slab pa lahko sposobne in prizadevne razvijalce pripelje do napačnih rezultatov. Vse agilne metode imajo tudi načine za čim boljši prenos znanj med razvijalci.

Cilj projekta mora biti delujoča programska oprema. Dokumentira naj se le tisto, kar je kasneje v pomoč razvijalcem pri nadaljnjem razvoju ali kar potrebujejo uporabniki za učinkovito uporabo programskega paketa. Prizadevanje za popolno dokumentacijo je pogosto časovno zelo potratno in razvijalce demotivira.

Osnova za projekt, tako pri klasičnih metodologijah kot pri agilnih, je pogodba z naročnikom. Razlika je v tem, da mora biti pogodba pri agilnih metodologijah napisana manj natančno, opisovati mora le uporabniške zgodbe oziroma želje (angl. *User stories*), in mora vsebovati določilo, da naročnik oziroma bodoči uporabnik dejavno sodeluje pri projektu, saj najbolje ve, kaj natančno želi. Agilna metodologija XP kar najbolje vpelje takšen koncept sodelovanja z zapovedjo, da mora biti pri razvoju projekta fizično prisoten en predstavnik naročnika na določeno število razvijalcev. Vse agilne metodologije določajo redno dobavo delujočega, vendar še ne dokončanega programskega paketa naročniku, da lahko ta že zgodaj v ciklu razvoja ugotovi, ali razvoj vodi natančno v smer, kakršno si je zamislil.

Spremembe želja naročnika pri razvoju programske opreme so najšibkejša točka tradicionalnih metodologij. Z možnostjo spreminjanja želja naročnika med potekom, nudijo agilne metodologije naročniku konkurenčno prednost. Projekti razvoja programske opreme pogosto trajajo več let in naročnikovo poslovno okolje in z njim potrebe se lahko medtem zelo spremenijo, enako kot tehnologije in zakonodaja. Agilne metodologije tako uporabljajo natančno načrtovanje le za krajše obdobje (na primer od enega tedna do največ dveh mesecev) in naročnik ima možnost spremeniti v pogodbi navedeno uporabniško zgodbo, ki še ni izvedena in ni v kratkoročnem načrtu.

2.2.2 Osnovni principi agilnih metodologij

Vse štiri vrednote, navedene zgoraj podpira dvanajst principov agilnih metodologij. Nekateri so omenjeni že pri razlagi posamezne vrednote, spodaj pa so našteti vsi (Principles behind the Agile Manifesto, 2001):

- Najvišja prioriteta pri razvoju programske opreme je zadovoljiti naročnika s hitro in pogosto dobavo uporabne programske opreme;
- Dobava programske opreme v kratkih časovnih intervalih: od enega tedna do nekaj mesecev, pri čemer se priporoča krajše časovno obdobje;
- Spremembe zahtev naročnika je treba sprejeti v vsakem trenutku, čeprav pozno v življenjskem ciklu razvoja programske opreme. Agilni razvoj izkoristi dejstvo, da naročniku omogoča spreminjanje zahtev, za naročnikovo konkurenčno prednost;
- Naročniki projekta in razvijalci morajo biti ves čas razvoja v stiku;
- Projekt naj temelji na motiviranih posameznikih. Razvojni ekipi je treba dati okolje in podporo, ki jo potrebuje, in ji zaupati, da bo svoje delo opravila kakovostno;
- Najučinkovitejša in najuspešnejša metoda za prenos informacij je neposreden pogovor;
- Uspešna programska oprema je osnovno merilo za uspešnost ekipe;
- Agilne metodologije se zavzemajo za vzdržen in nenehen razvoj. Predvideni tempo razvoja mora biti takšen, da so ga naročniki, razvijalci in uporabniki sposobni ohraniti;
- Zagotovljena morata biti neprestana pozornost glede tehnične odličnosti in dober načrt, kar veča agilnost razvoja;
- Preproste rešitve in izločanje nepotrebnega dela so ključnega pomena;
- Samo organizirane razvojne skupine izdelajo najboljše zahteve, načrte in arhitekturne rešitve;
- Razvojna ekipa v rednih časovnih presledkih ugotavlja, kako izboljšati svojo učinkovitost, in sprejema ustrezne ukrepe.

2.2.3 Vpliv principov agilnih metodologij na kakovost izdelka

Definicija kakovosti, kakršno navaja Gregory (2010), da je kakovosten tisti produkt, ki doseže oziroma preseže naročnikova oziroma kupčeva pričakovanja, je cilj razvoja z agilnimi metodologijami. K temu cilju prispeva sleherni, v prejšnjem poglavju navedeni princip agilnih metodologij. Čeprav se agilne metodologije med seboj razlikujejo in lahko ponudijo različne rešitve za nekatere od zgoraj naštetih principov, je cilj vedno enak. Spodaj je navedeno, kako posamezni principi prispevajo h kakovosti končnega izdelka (povzeto po virih: Cohn, 2010; Beck, 1999; Poppendick, 2010):

- Prvi princip poudarja, naj naročnik čim prej dobi določeno verzijo programskega paketa. Prej ko naročnik vidi določene funkcionalnosti, prej se lahko odzove, če si je določene rešitve predstavljal drugače. Glede na to, da naročnik postavi tudi prioritete zahtevkom,

ni dvoma, da bo najpomembnejše dele programskega paketa dobil v skladu s pričakovanji že pred rokom. Ker naročnik prejme pomembne dele programskega paketa zgodaj v življenjskem ciklu razvoja izdelka, imajo uporabniki dovolj časa, da jih spoznajo in jih začnejo takoj uporabljati.

- Drugi princip, ki poudarja dobavo novih različic programskega paketa v rednih in kratkih časovnih presledkih, ima podobno vlogo kot prvi, le da poteka skozi celotni razvojni cikel. Redno prejemanje programskega paketa dobro vpliva tudi na naročnikovo zaupanje in njegovo pripravljenost za sodelovanje z razvijalci.

- Tretji princip prinese naročniku zelo veliko dodano vrednost. Nič mu namreč ne koristi zelo kakovostno izdelana funkcionalnost, ki je ne potrebuje, saj so se v času projekta spremenile njegove poslovne okoliščine oziroma se je povečalo njegovo razumevanje delovanja programske opreme. Možnost sprememb med projektom projekta občutno povečuje kakovost izdelka po zgornji definiciji.

- Stik med naročnikom in razvijalci med potekom projekta zagotavlja, da se izdelek razvija v pravo smer, hkrati pa naročnika sproti seznanja s trenutnim stanjem projekta. V primeru, da izdelka ni mogoče izdelati v celoti v roku, je za naročnika pogosto zelo pomembno, da to izve čim prej.

- Primerno okolje za razvijalce ustvarjajo številni bolj ali manj pomembni dejavniki, ki vplivajo na ustvarjalno ozračje. To je na primer dovolj velik prostor za razvijalce, da lahko vsi delajo v enem prostoru, kar olajša komunikacijo v živo. Potrebna so primerna orodja, tako programska kot strojna, da omogočijo avtomatsko testiranje. Potrebna je podpora drugih oddelkov v podjetju, na primer kadrovskega, saj neprimerni načini nagrajevanja posameznikov lahko porušijo sodelovanje v ekipi. Takšno delovno okolje motivira posameznike, da naredijo kakovosten izdelek.

- Neposredna oblika pogovora je najbolj informativna oblika prenosa informacij, saj poleg besed obsega še kretnje posameznikov, mimiko in intonacijo govora, kar vse prispeva k popolnosti podane informacije. Proces, ki omogoča in spodbuja neposredno obliko pogovora, skrbi, da ne prihaja do napačne interpretacije vsebin, ki jih mora posameznik opraviti. Slabost neposredne oblike pogovora je, da je ni mogoče shraniti za prihodnje potrebe, zato je treba pomembne točke dogovorjenega zapisati, vendar je taka dokumentacija bolj zgoščena, kar prihrani nepotrebno dokumentiranje in lajša kasnejše iskanje dogovorjenega.

- Princip merila uspešnosti ekipe se zdi samoumeven, vendar ga le redka podjetja izpeljejo v praksi. Vsem v ekipi mora biti jasno, da bodo primerno nagrajeni le, če bo celotna ekipa uspešno izvedla projekt. Načini nagrajevanja, ki so pogosti pri tradicionalnih metodologijah, so v nasprotju s tem principom. Kadar je posamezni programer nagrajen

glede na napisane vrstice kode ali število napisanih metod, to negativno vpliva na kakovost izdelka, saj jo agilne metodologije zagotavljajo s preprosto in lahko berljivo kodo s čim manj podvajanja. Ravno tako testerjeva nagrada ne sme biti odvisna od števila najdenih napak, saj je agilni proces razvoja določen tako, da razvijalci ne naredijo napak pri programiranju, odgovorni za kakovost pa jim pri tem pomaga. O načinu nagrajevanja posameznikov je treba zelo dobro razmisliti, da bo razvijalce spodbujal k pravemu cilju in večji kakovosti izdelka.

- Princip vzdržnega in konstantnega razvoja opozarja, da razvijalci ne smejo delati nadur, vsaj ne dva tedna zapored. Agilni razvoj predvideva konstantno dostavljanje programskega paketa, zato mora biti delo razporejeno približno enakomerno. Nadurno delo v eni iteraciji je dopustno, nadurno delo v več zaporednih iteracijah pa opozarja na napake v procesu. Pretirana obremenitev veča napake pri delu zaradi manjše osredotočenosti razvijalcev in upadanja morale.

- Princip neprestane skrbi za tehnično odličnost zahteva neprestano skrb za kakovost, saj od razvijalcev zahteva, da redno spreminjajo slabe, odvečne dele kode (angl. *refactoring*) in ne uporabljajo bližnjic zato, da bi ujeli roke.

- Sposobnost ugotoviti, česa ni treba narediti, je lahko ključnega pomena, posebej kadar nas priganjajo roki za izdelavo produkta. Prihranjeni čas uporabimo za kakovostno izdelavo pomembnih delov programske opreme.

- Pri agilnih metodologijah ima vodstvo le svetovalno vlogo in mora razvojni ekipi zaupati. Razvojno ekipo zadolži za izvedbo določenega sklopa, način kako izvesti sklop pa ostane v pristojnosti ekipe. Tako se ekipa kot celota počuti odgovorno za izvedbo kakovostnih rešitev.

- Ker noben proces ni popoln, agilne metodologije predvidijo načine, kako lahko ekipa sama, med procesom razvoja izpopolni lastni proces dela.

2.2.4 Vrste agilnih metodologij

Različne agilne metodologije, ki imajo skupno filozofijo in vrednote, imajo podobne značilnosti in pogosto zahtevajo enake dobre prakse dela, razlikujejo pa se v izvedbi, naboru dobrih praks in zahtevnosti. Nobena od metodologij ni univerzalna, zato se mora podjetje odločiti, katera je najprimernejša glede na okoliščine, v katerih dela. Včasih v posameznem podjetju uporabljajo na različnih projektih različne metodologije, čeprav je takšna praksa smiselna predvsem pri uvajanju nove metodologije (Kniberg, 2010).

Najpogosteje se uporablja metodologija **Scrum**, ki ne predpisuje specifičnih tehnik razvoja, ampak je namenjena predvsem vodenju projekta. Podrobno je opisana v poglavju 2.2.5.

Veliko bolj pa tehnični vidik poudarja razširjena metodologija **XP** – ekstremno programiranje (angl. *Extreme Programming*), ki se jo tudi pogosto uporablja v kombinaciji z metodologijo Scrum. Metodologija XP je opisana v poglavju 2.2.6.

Manj zahtevna od Scruma, vendar izpeljana iz njega, je metodologija **Kanban**, ki postaja priljubljena v zadnjem času. Kanban večino lastnostih prevzema iz Scruma. Posebna značilnost te metodologije je, da mora ekipa določiti, koliko je lahko hkrati nalog v določeni fazi razvoja, zato je primerna za projekte »Just in Time« (Kniberg, 2010).

Najlažja med agilnimi metodologijami je družina metodologij **Crystal**. Njihova posebnost je, da podjetje glede na zahtevnost, velikost ekipe in prioriteto izbere eno izmed med seboj podobnih metodologij Crystal (Crystal Clear (software development), 2010).

Dinamična metodologija za razvoj sistemov (angl. *Dynamic Systems Development Method*) – **DSDM**, je metodologija, ki zahteva intenzivno sodelovanje naročnika, saj je njen glavni cilj izdelati izdelek v roku in v okviru proračuna, medtem ko je nabor funkcionalnosti lahko spremenljiv. Ta metodologija je primerna za kombiniranje s kako drugo metodologijo, ki ima več tehničnih zahtev, na primer XP (Dynamic Systems Development Method, 2010).

Funkcijsko vodeni razvoj (angl. *Feature-Driven Development* – v nadaljevanju **FDD)** je metodologija, ki predvideva izvedbo programskega paketa z izvedbo posameznih značilnosti, tako da se vsako značilnost zaporedno načrtuje, implementira in dobavi naročniku (Feature Driven Development, 2010).

Vitek razvoj programske opreme (angl. *Lean Software developement* – v nadaljevanju **LSD)** je metodologija, ki si prizadeva za produktivnost in da naročnik najprej dobi tiste funkcionalnosti, ki imajo zanj največjo dodano vrednost. Primeren je tudi za razvoj zelo velikih projektov, ki jih ni mogoče vnaprej načrtovati, saj vse bodoče funkcionalnosti še niso znane (Poppendieck, 2007).

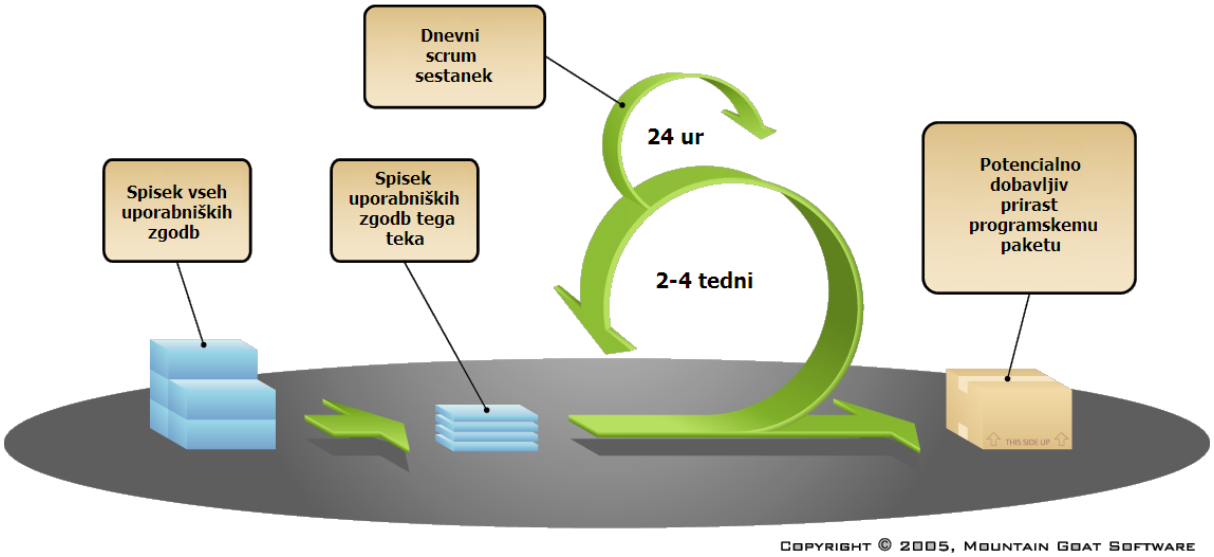
2.2.5 Scrum

Metodologijo Scrum so razvili za uspešnejše upravljanje projektnega dela in ne določa specifičnih tehnik pri razvoju, zato ni naravnana izključno na IT industrijo. Poudarja prilagodljivost, prožnost in ustvarjalnost, s čimer omogoča agilni razvoj projekta v stalno spreminjajočem se okolju zahtev.

Zgodovina razvoja te metodologije se je začela z letom 1986, ko so za hiter in prilagodljiv razvoj novih izdelkov razvili nov holističen pristop. V začetku so ga primerjali s taktiko, kakršno uporabljajo v ameriškem nogometu, ko si ekipa s kombinacijo podaj prizadeva doseči cilj. Tak pristop k razvoju izdelka so leta 1991 v nekem članku prvič imenovali Scrum (2010).

V praksi pa sta Scrum prvič uporabila Ken Schwaber in Jeff Stuherland, vsak v svojem podjetju. Predstavila sta ga na konferenci leta 1995. Leta 2001 sta Schwaber in Mike Beedle razložila metodologijo Scrum v knjigi Agile Software Development with Scrum, ki je izhodišče za vrsto literature na temo razvoja z metodologijo Scrum (Schwaber & Beedle, 2001).

Slika 5: Grafični prikaz poteka iteracije z metodologijo Scrum



Vir: Mountain Goat Software, 2010.

Ogrodje metodologije Scrum deluje po načelu »nadziraj in prilagodi« (angl. *inspect and adapt*). Sestavljeno je iz treh vlog, treh obredov in treh dokumentov. Tako načrtovano ogrodje dostavlja izdelek v tekih (angl. *Sprints*). Na Sliki 5 je prikazan potek enega teka, ki so navadno enomesečne (od enega tedna do največ dveh mesecev) iteracije (Kniberg, 2007).

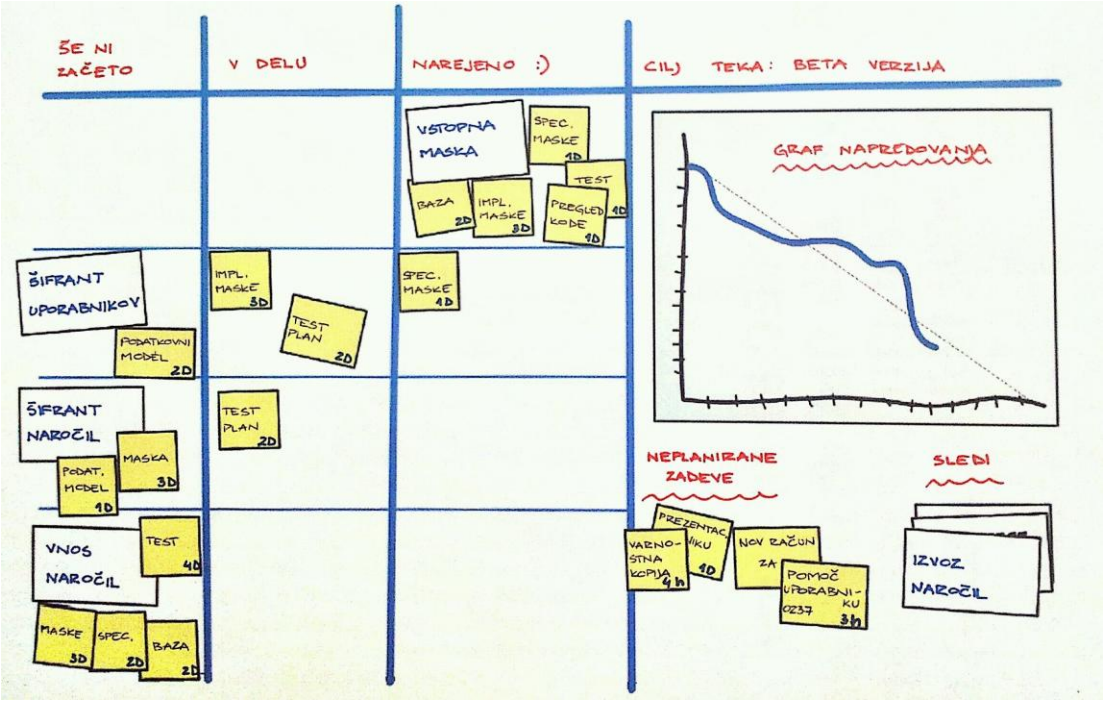
Ogrodje metodologije Scrum (Kniberg, 2007):

- **Vloge** (angl. *Roles*) so: Naročnik (angl. *Product owner*), skrbnik procesa (angl. *Scrum Master*) in ekipa (angl. *Scrum Team*).
- **Obredi** (angl. *Ceremonies*) so: Planiranje teka (angl. *Sprint planning*), pregled teka (angl. *Sprint Review*) in dnevni Scrum (angl. *Daily Scrum meeting*).
- **Dokumenti** (angl. *Artifacts*) so: Spisek vseh uporabniških zgodb (angl. *Product Backlog*), spisek uporabniških zgodb, planiranih za naslednjo iteracijo (angl. *Sprint Backlog*) in graf napredovanja (angl. *Burn down Chart*).

Naročnik je predstavnik bodočih uporabnikov. Njegova pristojnost je, da določa uporabniške zgodbe (angl. *User stories*), na Sliki 5 so prikazane z modro, njihovo prioriteto in da po vsakem prejemu različice izdelka določi, ali je bil cilj iteracije (angl. *Sprint goal*) dosežen. Odgovoren je tudi za donosnost projekta. Skrbnik procesa skrbi, da ekipa ne zaide s predpisane poti, vodi Scrum obrede in pomaga ekipi, kadar nastopi kak problem. Odgovoren

je torej za uveljavljanje načela »nadziraj in prilagodi«. Na začetku projekta ekipa v večji meri obremeni skrbnika procesa, proti koncu projekta pa se obremenjenost prevesi na stran naročnika. Ekipa (2 do 20 razvijalcev) je odgovorna za analizo, načrt, razvoj, testiranje in dostavo nalog, določenih v posameznem teku.

Slika 6: Tabla z osnovnimi podatki o stanju napredka v trenutni iteraciji



Vir: H. Kniberg, Scrum and XP from Tranches, 2007.

Plan teka je daljši sestanek (nekaj ur) na začetku vsakega teka, na katerem ekipa določi uporabniške zgodbe, ki jih bo implementirala v sledečem teku. Pri izboru uporabniških zgodb upošteva prioritetni red, ki ga je uporabniškim zgodbam določil naročnik. Ekipa skupaj z naročnikom določi cilj posameznega teka, primer je narisan desno zgoraj na Sliki 6. Uporabniškim zgodbam, razčlenjenim na posamezne naloge (angl. *tasks*), ki so na Sliki 6 prikazane z rumenimi listki, ekipa določi število točk zahtevnosti (angl. *effort*). Iz izkušenj iz prejšnjih iteracij ekipa ve, koliko točk zahtevnosti je sposobna uresničiti v posameznem časovnem obdobju, zato določi, katere uporabniške zgodbe ali le njihove dele (naloge) lahko umesti v naslednji tek. Pregled teka je sestanek, na katerega so poleg naročnika povabljeni tudi bodoči uporabniki izdelka, ki skupaj določijo, ali je bil cilj posameznega teka dosežen. Pregled teka opravijo po dobavi posamezne različice izdelka, ko se naročnik že seznani z novostmi. Dnevni Scrum je vsakodnevni sestanek, katerega namen je oceniti, kaj je ekipa dosegla od prejšnjega dnevnega Scrum sestanka, kaj bo realno dosegla do naslednjega sestanka in razjasniti morebitne ovire razvijalcev pri doseganju dnevnega cilja.

Spisek vseh uporabniških zgodb oziroma zahtevkov, ki je na Sliki 5 označen z modrimi kvadri, je navadno izsek pogodbe med naročnikom izdelka in podjetjem, ki izdelek razvija. Del pogodbe, ki govori o vsebini implementiranega, je razčlenjen na uporabniške zgodbe. To

so enostavni, jasni in jedrnati opisi funkcionalnosti, ki bodo koristili končnim uporabnikom izdelka. Najbolje je, da je že pogodba med naročnikom in izvajalcem napisana v obliki uporabniških zgodb. Vsaki uporabniški zgodbi v spisku mora biti določen prioritetni red, da lahko se lahko ekipa razvijalcev najprej loti bolj nujnih delov izdelka. Spisek uporabniških zgodb, planiranih za naslednji tek, je na Sliki 5 označen z modrimi lističi. Sestavljen je iz spiska vseh zahtevkov tako, da bo izdelek v naslednji različici za naročnika postal čim koristnejši. Sestavi ga ekipa razvijalcev, saj lahko ta najbolj realno oceni, kaj je sposobna narediti v enem teku. Prikaz spiska uporabniških zgodb, razčlenjenih na posamezne naloge in razporejenih glede na stanje posamezne naloge, je na levi strani Slike 6 približno na sredini iteracije. Graf napredovanja, ki prikazuje upadanje števila še nedokončanih nalog v času posamezne iteracije je na Sliki 6 prikazan na desni strani table.

Scrum kot agilna metodologija se drži principa, da se dokumentira le tisto, kar je nujno za razvoj izdelka, kasnejšo podporo in za razumevanje izdelka s strani naročnika. Ker opisani trije dokumenti niso potrebni iz teh razlogov, ampak so le v pomoč pri vodenju projekta in določanju prioritet, večinoma niso narejeni v obliki besedilnega dokumenta, ki ga je mogoče shraniti in natisniti. Kot je prikazano na Sliki 6, so uporabniške zgodbe navadno napisane le na lističe, ki se jih prilepi na tablo, graf napredovanja pa je ravno tako pogosto narisan na tabli, ki je na vsem razvijalcem na očeh. Takšen način dela odpravlja nepotrebno izgubo časa s posodabljanjem dokumentacije. Seveda obstaja več različnih orodij za podporo vodenju z metodologijo Scrum, ki zgornje dokumente digitalizirajo, vendar strokovnjaki za to metodologijo večinoma odsvetujejo rabo digitaliziranih pripomočkov, če ni posebnih razlogov zanje.

Metodologija Scrum z opisanimi obredi in dokumenti zagotavlja kakovost izdelka. Proces razvoja vsebuje neposredne komunikacijske kanale, tako med razvijalci kot med naročnikom in ekipo. Metodologija zagotavlja, da ekipa pri svojem delu nenehno napreduje in odstranjuje nepotrebne elemente, ki ne prispevajo h kakovosti izdelka ali učinkovitejšemu razvoju.

V primerjavi z drugimi agilnimi metodologijami se Scrum bolj poglablja v vodenje procesa razvoja, redno dostavo končanih delov aplikacije naročniku in njegovo sodelovanje pri določanju prioritet. Ker Scrum ne določa posebnih tehnik pri sami implementaciji, je to metodologijo priporočljivo kombinirati s katero drugo, ki bolj poudarja tehnične plati razvoja, oziroma upoštevati dobre prakse drugih metodologij. S Scrumom pogosto kombinirajo metodologijo XP, ki je opisana v naslednjem poglavju. Tako poskrbijo za kakovostno izvedbo nalog, ki jih naročnik potrebuje.

2.2.6 Ekstremno programiranje

Ustanovitelj metodologije ekstremno programiranje – XP (angl. *Extreme Programming*) Kent Beck zagotavlja, da je XP lahek, učinkovit, prilagodljiv, napovedljiv in zabaven način razvoja programske opreme brez tveganja (Beck, 1999).

Beck K. je XP razvil, ko je bil zaposlen v podjetju Chrysler, kjer so razvijali nov programski paket C3, plačilni sistem za svojih 87.000 zaposlenih. Pri tem projektu C3 je začel uporabljati način dela, ki ga danes imenujemo XP. Od trenutka, ko je začel voditi projekt, do prve različice C3, je minilo približno eno leto, kar je bil velik uspeh. Kljub temu programski paket ni nikoli zares zaživel, saj po odhodu predstavnika naročnika podjetje ni našlo njegovega naslednika. Vloga predstavnika naročnika je namreč ključna za uspešen XP razvoj. Kent Beck je tudi eden od ustanoviteljev Agile Manifesta (Extreme_Programming, 2010).

Ekstremno programiranje, v nadaljevanju XP je sestavljeno iz različnih dobrih praks, ki jih uporabljajo v klasičnih metodologijah razvoja. Te so v XP včasih uporabljene do svoje skrajne meje in prvič povezane, tako da se med seboj čim bolj dopolnjujejo.

XP je sestavljen iz štirih sklopov. Najširše ga določajo **vrednote**, ki se ne nanašajo posebej na področje razvoja programske opreme, vendar upoštevanje teh vrednot zagotavlja učinkovito in kakovostno delo. Vrednote metodologije XP so pogum, komunikacija, preprostost, povratne informacije in spoštovanje. **Pogum** je potreben, da se pravilno odločimo, tudi takrat, ko je težko in potreben je pogum, da vsem udeleženim povemo resnico, če je nujno. **Komunikacija** je pomembna, da pravilna informacija doseže prave posameznike. Le tako jo je mogoče kar najbolj izkoristiti. **Preprostost** pomeni, da zavržemo, kar si želimo, a ne potrebujemo. **Povratne informacije** so pomembne, da se učimo iz vsake priložnosti. Potrebno je medsebojno **spoštovanje**, priznavanje strokovnega znanja drug drugemu in želja po skupnem uspehu (Schwaber & Beedle, 2001).

Principi, ki jih določa XP, da usmerijo vrednote na področje razvoja programske opreme, so hitre povratne informacije, princip enostavnosti, neprestano uvajanje majhnih sprememb, sprejemanje sprememb, kakovost dela in drugi, stranski principi (Beck, 1999).

Delovne prakse določajo, kako principe XP uveljaviti pri dejanskem delu na projektu. Prakse metodologije XP so igra načrtovanja, programiranje v parih, skupno lastništvo kode, majhne in časovno kratke verzije, uporaba podob za opis sistema, preprost model, testiranje, preoblikovanje kode, neprestana integracija, 40-urni delovnik, stalna dosegljivost naročnika in vzpostavljeni standardi kodiranja (Beck, 1999).

Delovne prakse so sestavljene iz posameznih **aktivnosti** pri delu: poslušanja, preverjanja, kodiranja in načrtovanja (Beck, 1999).

Ekstremno programiranje od vseh agilnih metodologij vpelje največ radikalnih metod dela. Uporaba XP daje kakovostne izdelke na učinkovit način le, če podjetje uporablja vse dobre prakse, ki jih predpisuje XP, saj se le te med seboj dopolnjujejo. Le redkokatero podjetje zbere dovolj volje in želje za uporabo vseh dobrih praks, ki jih predpisujejo XP, le delno vpeljana XP metodologija pa ne dosega pričakovanih rezultatov.

To metodologijo pogosto kritizirajo zaradi radikalnosti njenih dobrih praks. Najpogostejši so naslednji očitki (McBreen, 2003; Boehm, 1999; Stephens & Rosenberg, 2003):

- Premalo strukturiran proces in pomanjkanje potrebne dokumentacije;
- XP deluje samo z izkušenimi razvijalci;
- Načrtovanje ni dovolj natančno;
- XP metodologija zahteva uporabo vseh dobrih praks in, če podjetju spodleti pri vpeljavi ene, mu spodleti pri vpeljavi celotne XP metodologije;
- Čas, ki ga naročnik porabi pri sodelovanju v razvoju, je skriti strošek;
- Pogajanja o pogodbenih obveznostih so lahko še napornejša;
- Za vpeljavo metodologije XP je potrebna zelo velika sprememba kulture podjetja;
- V začetnih iteracijah je načrtovanje zelo težavno, saj ekipa še nima izkušenj, koliko je sposobna narediti v posamezni iteraciji;
- Agilni razvoj temelji na implementaciji funkcionalnosti, zato je nefunkcionalne zahteve težko vključiti v uporabniške zgodbe.

Čeprav so nekatere kritike upravičene (predvsem zadnje tri), za večino kritik zagovorniki agilnega razvoja in metodologije XP trdijo, da izvirajo iz nezadostne poučenosti o metodologiji (Schwaber & Beedle, 2001).

3 ISO STANDARD KAKOVOSTI

Standard ISO 9001:2008 je, kot pove že ime, izdala nevladna mednarodna organizacija ISO – International Standardisation Organization leta 2008. Gre za četrto izdajo standarda (druge so izdali v letih 1987, 1994 in 2000). Prenovljena izdaja se ne razlikuje bistveno od izdaje leta 2000, ki pa je naredila korenit obrat k učinkovitejšemu sistemu vodenja kakovosti (Hoyle, 2009).

Standard je namenjen podjetjem vseh tipov in velikosti, zgrajen pa je na načelih vodenja kakovosti, ki jih danes uporabljajo v svetu uspešna podjetja. Tako je standard povzetek dobrih poslovnih praks in kot tak v pomoč podjetjem, ki imajo za vzor samo najboljše (Hoyle, 2009).

Standard določa zahteve za (Slovenski inštitut za standardizacijo – SIST, 2011):

- **Sistem vodenja kakovosti** – Podjetje mora najprej opredeliti, kateri so njeni procesi, kako ti delujejo drug na drugega, kateri viri so potrebni, da nastane izdelek, in kako bo procese merilo in izboljševalo. Nato mora s poslovnikom kakovosti in nadzorom zapisov vzpostaviti še sistem za obvladovanje dokumentacije;
- **Odgovornost vodstva** – Najvišje vodstvo v podjetju se mora dobro zavedati tega pomembnega dela standarda. Vodstvo je namreč odgovorno za določanje politike in

ciljev ter za pregled sistemov, hkrati pa tudi za obveščanje o učinkovitosti sistema znotraj podjetja;

- **Vodenje virov** – Novi standard bolj poudarja vire, ki si jih mora podjetje zagotoviti, da bo odjemalec dobil tisto, kar so se dogovorili. Sem ne spadajo samo ljudje, temveč tudi fizični viri, kot so oprema, prostori in vse potrebne pomožne storitve;
- **Realizacija proizvoda** – Ta del sestavljajo procesi, ki so potrebni za izvedbo izdelka oziroma storitve. K takim procesom spadajo dejavnosti, kot je sprejemanje navodil od odjemalcev, snovanje in razvoj proizvodov, nabava materiala in storitev ter dobava izdelkov in storitev;
- **Merjenje, analize in izboljševanje** – Nadzorovanje in merjenje proizvodov, procesov, zadovoljstva odjemalcev in sistema vodenja ter zagotavljanje stalnega izboljševanja sistema so za vodenje sistema bistveni.

Izpolnjevanje vseh zahtev standarda, ki bodo podrobneje opisane v četrtem poglavju, omogoča podjetju, da po uspešno zaključenem certifikacijskem postopku pridobi certifikat za sisteme vodenja kakovosti po ISO 9001:2008. To prinaša podjetju številne prednosti.

3.1 Prednosti in slabosti certifikacije ISO 9001 za podjetje

Število podjetij, ki so pridobila certifikat ISO 9001, je začelo naglo naraščati po letu 1993, ko je bilo certificiranih manj kot 50.000 podjetij. Leta 2009 je imelo ta certifikat že več kot milijon podjetij na svetu (Internal Organization for Standardization, 2009b). Sodeč po tem se je v svetu pojavilo prepričanje, da certifikat ISO 9001 prinese podjetjem veliko prednosti (Heras, Dick & Casadesús, 2002).

Raziskave dokazujejo, da so podjetja, ki so skladna s certifikatom ISO 9000 v povprečju donosnejša, kar pa ne pomeni, da skladnost s certifikatom že sama pomeni večjo donosnost podjetja. Za certificiranje in s tem skladnost s certifikatom se namreč v večji meri potegujejo boljša podjetja, ki so bolj donosna že pred samim postopkom certifikacije (Heras, Dick & Casadesús, 2002).

Podjetja se za certifikacijo ISO standarda kakovosti odločijo zaradi različnih vzrokov, zunanjih ali notranjih. Značilna zunanja vzroka sta (Nanda, 2003):

- Standardi omogočajo podjetju, da se prijavi na razpise, ki od dobaviteljev zahtevajo, da imajo registracijo ISO 9001. Pogosti so tudi primeri, ko naročnik postavi potencialnemu dobavitelju vprašalnik o formalnem sistemu vodenja kakovosti. V takšnih primerih ima podjetje, skladno z ISO 9001 konkurenčno prednost, saj mu certifikat omogoča oziroma olajšana prijavljanje na razpise in pridobivanje novih projektov.
- Dejstvo, da je sistem vodenja kakovosti podjetja preveril zunanji opazovalec, vliva obstoječim in potencialnim strankam občutek varnosti in zaupanja.

Notranji vzrok za certifikacijo ISO 9001 je na primer, zavedanje podjetja, da mora narediti korak naprej v načinu poslovanja. Prenovljeni standard ISO 9001:2008 namreč posebej poudarja usmerjenost k odjemalcu, kar je prvi pogoj za donosno poslovanje. Podjetje mora vzpostaviti takšen odnos z odjemalcem, da je v obojestransko korist. Med prednostmi standarda kakovosti ISO 9001:2008 lahko omenimo še da (SIST, 2011):

- je prožen sistem vodenja;
- temelji na procesu in ne na postopkih;
- spodbuja stalno izboljševanje;
- uporabi zadovoljstvo odjemalca kot merilo za uspešnost sistema vodenja kakovosti;
- vsakogar motivira s skupnim ciljem in zagotavlja sodelovanje;
- v širokem obsegu vključuje najvišje vodstvo, saj poslovne odličnosti ni mogoče delegirati;
- se navezuje na zakonske in regulatorne zahteve;
- zahteva postavitev izmerljivih ciljev na različnih ravneh sistema, funkcije in proizvoda;
- se osredotoča na učinkovito notranje komuniciranje;
- usmerja pozornost na razpoložljive vire;
- zahteva vrednotenje učinkovitosti usposabljanja in vodenja kakovosti;
- je bolj jasno pri samih zahtevah (besedilo prejšnjega standarda ni bilo dovolj jasno) in povečanje usklajenosti z SIST EN ISO 14001:2005.

Čeprav so prednosti skladnosti s ISO standardom kakovosti nesporne, kritiki opozarjajo na šibke točke:

- Strošek za podjetje ob certificiranju podjetja z ISO 9001 se pogosto zelo dolgo časa ne povrne (Stephanie, 2005);
- Če podjetja privzamejo pridobitev certifikata ISO 9001 za svoj osnovni cilj, lahko skladnost s certifikatom podjetju sicer pomaga doseči določeno raven kakovostnega in organiziranega dela, vendar podjetja ne sili v napredovanje. Podjetja si namreč sama postavljajo cilje za napredovanje (Stephanie, 2005);
- Certificiranje ISO 9001 pogosto sloni na preverjanju, kako podjetje izpolnjuje kupčeve zahteve, ne sili pa ga v interno izboljšanje kakovosti izdelkov (Barens, 2000).

3.2 Oris ISO 9001:2008

ISO standard kakovosti temelji na modelu PDCA – Planiraj-Izvedi-Preveri-Ukrepaj (angl. *Plan-Do-Check-Act*), ki je prikazan s Sliko 7. PDCA predstavlja postopek, ko v podjetju najprej dokumentirajo, kar želijo narediti (skladno z zahtevami naročnika in primerno standardu), šele nato sledi izvedba planiranega. Ko so plani uresničeni, v podjetju preverijo in zabeležijo, da je bila izvedba narejena po planu. Če ni, je treba ukrepati, da bo naslednjič bolje. Če hoče podjetje registracijo ohraniti, je potrebna revizija vsaka tri leta in mini revizija

vsakega pol leta. Takrat podjetje dokaže, da deluje po modelu PDCA (Internal Organization for Standardization, 2011b).

Slika 7: Grafikon PDCA (Planiraj – Izvedi – Preveri – Ukrepaj)

Vir: povzeto po Slovenski inštitut za standardizacijo – SIST, Sistem vodenja kakovosti in slovenski standard sist en ISO 9001:2008, 2011.

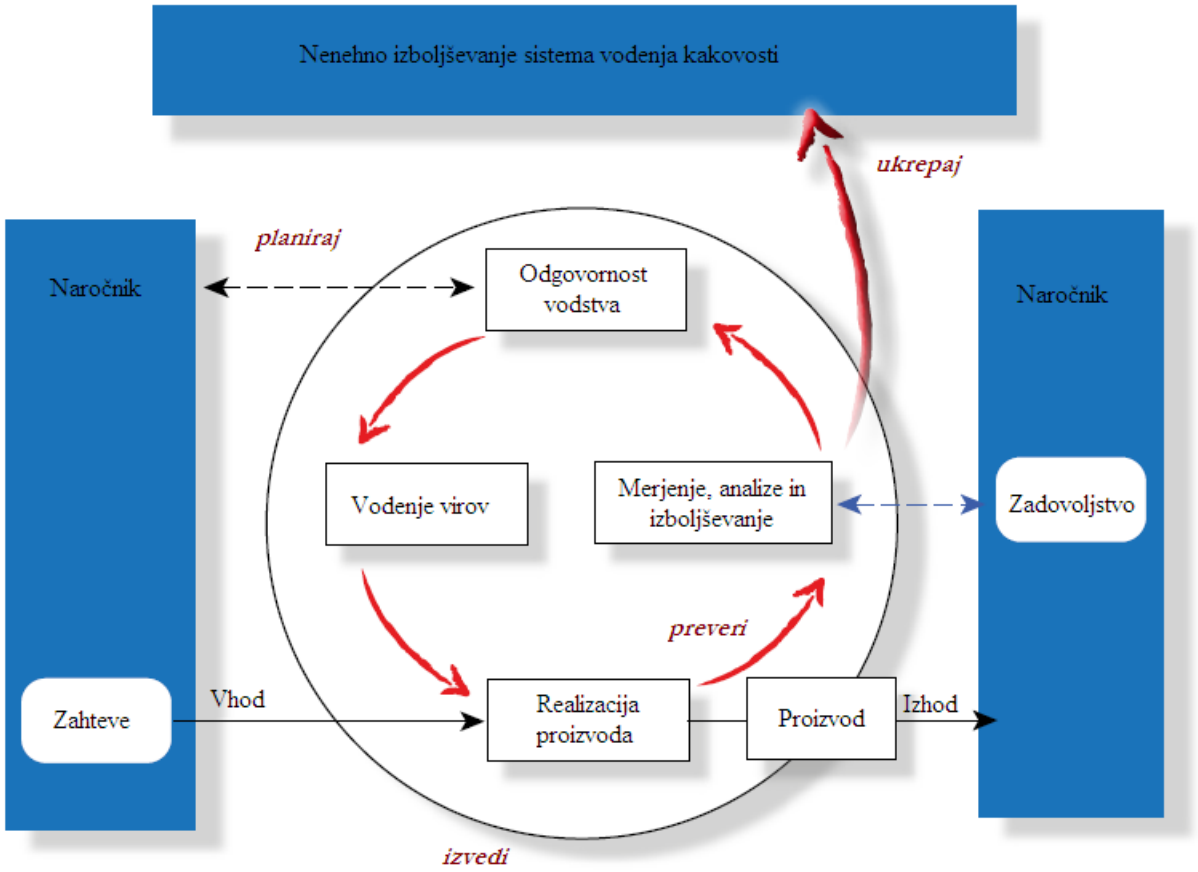
Vsebina sistema vodenja kakovosti po ISO 9000:2008 temelji na osmih načelih. Ta so sad znanja in izkušenj mednarodnih strokovnjakov, ki sodelujejo v ISO Technical Committee. Ta načela so (Internal Organization for Standardization, 2011a):

- **Osredotočenost na odjemalca** – Podjetje je odvisno od svojih odjemalcev, zato mora razumeti njihove trenutne in bodoče potrebe, zadovoljiti njihove zahteve in si prizadevati za preseganje odjemalčevih pričakovanj;
- **Skrb za kakovostno vodenje** – Vodje morajo skladno s ciljem določiti smer, v katero se giblje podjetje. Vzpostaviti morajo delovne pogoje, da lahko sodelavci izpolnijo cilje podjetja;
- **Udeležba ljudi** – Človek je jedro podjetja na vseh ravneh. Podjetje lahko v celoti izkoristi njegove sposobnosti le, če je popolnoma prisoten;
- **Procesna naravnanost** – Podjetje učinkoviteje doseže želeni rezultat, če vse aktivnosti in povezane vire upravlja procesno;
- **Sistemski pristop k vodenju** – Uspešnost in učinkovitost podjetja sta odvisni od sistematičnega pristopa do delovnih opravil;
- **Neprestano izboljševanje** – Neprestano izboljševanje učinkovitosti delovanja mora biti trajni cilj podjetja;
- **Odločitve morajo temeljiti na dejstvih** – Učinkovite odločitve temeljijo na logičnih analizah podatkov in preverjenih informacijah;
- **Vzajemno koristni odnosi med odjemalci in dobavitelji** – Podjetje mora gojiti vzajemno koristne odnose s svojimi dobavitelji, kar je dobro za oba.

Načela ISO standarda kakovosti in model planiraj-izvedi-preveri-ukrepaj (v nadaljevanju PDCA) so skupaj uporabljena v procesnem modelu za doseganje neprestanega izboljševanja

kakovosti. Ta model je prikazan na Sliki 8 in je jedro standarda ISO 9001:2008 (Internal Organization for Standardization, 2009a).

Slika 8: Prikaz toka procesa po ISO 9001



Vir: Internal Organization for Standardization, ISO 9000 – Selection and use, 2009a.

3.3 Drugi standardi in vodila pri razvoju in vzdrževanju programske opreme

ISO 9001:2008 je edini ISO standard s področja kakovosti, za katerega je mogoče pridobiti certifikat o skladnosti. Pri uvajanju ISO 9001:2008 v podjetje, ki razvija in vzdržuje programsko opremo, pa so za lažje razumevanje standarda zanimivi še drugi standardi.

ISO 90003:2004 podaja navodila za interpretacijo zahtev ISO 9001 v postopku nakupa, prodaje, razvoja, obratovanja in vzdrževanja programske opreme ter pri opravilih, ki so s temi postopki povezani. Ta standard ne dodaja niti v ničemer ne spreminja zahtev ISO 9001:2000. Čeprav temelji na verziji standarda ISO 9001:2000, je primeren za uporabo tudi pri uvajanju standarda ISO 9001:2008, saj med njima ni velikih razlik (Pivka & Košir, 2010).

Ob branju ISO 90003:2004 je zelo primerna uporaba standarda **ISO/IEC 12207.** Na slednjega se namreč večkrat sklicuje ISO 90003:2004 pri definicijah različnih postavk. Standard ISO/IEC 12207 opisuje življenjski ciklus razvoja programske opreme. Definira, ne da bi določal podrobnosti, 22 procesov, 95 aktivnosti, 325 opravil in 254 ciljev v življenjskem ciklu razvoja programske opreme. ISO 90003:2004 se pogosto nanaša tudi na **ISO/IEC 15939**, ki je namenjen merjenju procesov pri razvoju programske opreme, in na **ISO/IEC 15504**, ki podaja ogrodje za oceno procesa razvoja (Pivka & Košir, 2010).

Ocenjevanju kakovosti programske opreme je namenjen standard **ISO/IEC 25000:2005**. Standard predstavlja model za ugotavljanje notranje kakovosti (ocenjevana na ravni posameznega izdelka), zunanje kakovosti (ocenjevana na ravni večjega sistema) in kakovosti programske opreme v uporabi (ocenjevanje kakovosti na ravni poslovnega sistema). Standard ISO/IEC 25000:2005 nadomešča (oziroma bo nadomestil, ko bo izpopolnjen) standarda ISO/IEC 9126 in ISO/IEC 14598, na katera se nanaša tudi ISO 90003:2000 (Pivka & Košir, 2010).

Alternativa ISO 90003:2004 je komercialna **TickIT shema** akreditacije in certifikacije, razvita v Angliji. Namenjena je implementaciji ISO 9001 na področju razvoja in vzdrževanja softvera. Drugače od ISO 90003:2004, ki le razlaga zahteve ISO 9001 pri razvoju in vzdrževanju programske opreme, pa TickIT shema podaja še konkretna navodila in primere, kaj je treba narediti, da podjetje izpolni določeno zahtevo. Za registracijo ISO 9001:2008 po TickIT shemi morajo podjetje revidirati presojevalci iz podjetja TickIT, kar da registraciji še večjo veljavo, saj so procese v podjetju revidirali izvedenci za na razvoj programske opreme. Negativna stran registracije ISO 9001 po TickIT shemi je gotovo cena, saj morajo podjetje revidirati strokovnjaki iz Anglije, kar povečuje potne stroške revizorjev (Pivka & Košir, 2010).

4 ZDRUŽLJIVOST ISO 9001 IN AGILNEGA RAZVOJA PROGRAMSKE OPREME

Vprašanje o tem, kako združiti agilni razvoj programske opreme z ISO 9001, je verjetno staro ravno toliko kot agilni razvoj programske opreme sam. Ustanovitelji agilnega manifesta te tematike sicer niso predvideli kot problem, pač pa so jo tako videli vodstveni kader in inženirji za zagotavljanje kakovosti.

V prvem delu tega poglavja, ki govori združevanju agilnega razvoja in ISO 9001, predstavljam pregled raziskav o tej tematiki. V drugem delu poglavja pa je razčlenjeno po točkah standarda prikazano, kako podjetje z uporabo agilnih metodologij razvoja programske opreme izpolnjuje zahteve posameznih točk standarda, katere prilagoditve je potrebno izvesti za izpolnjevanje zahtev in katere točke je teže izpolniti, da v podjetju ne izgubijo agilnosti razvoja pri posameznih projektih. V tretjem delu tega poglavja na kratko povzemam še ugotovitve združljivosti ISO 9001 in razvoja z agilnimi metodologijami v podjetju.

4.1 Pregled raziskav o združljivosti agilnega razvoja programske opreme in ISO 9001

Raziskave avtorjev o kompatibilnosti agilnega razvoja programske opreme in ISO 9001 so si med seboj zelo različne in različno uporabne za bralca. V tem poglavju predstavljam zanimivejše raziskave. Nekatere imajo dokaj ekstremna stališča, na primer, da takšno združevanje ni problematično ali da sploh ni potrebno, ker striktno zagovarjajo samo enega od obeh pogledov na kakovost. Drugi gledajo na združevanje dokaj inovativno, tretji so se tematike lotili bolj strukturirano in navajajo tiste točke standarda, ki združevanje otežujejo.

Najbolj sistematično se tematike združevanja ISO 9001 in agilnih metodologij lotita avtorja Stålhane in Hanssen. Avtorja povežeta zahteve standarda ISO 9001 in napotke za uporabo teh zahtev pri razvoju in vzdrževanju programske opreme (ISO 90003) z rešitvami, ki so rezultat uporabe dobrih praks agilnih metodologij. Na tak način prideta do ugotovitve, da je od 50 zahtev ISO 9001 standarda 31 zahtev avtomatsko implementiranih z doslednim sledenjem dobrih praks agilnega razvoja, 15 zahtev je le delno izpolnjenih, štirih zahtev pa agilne metodologije ne morejo izpolniti in jim je potrebno nameniti posebno pozornost. Pregled 19 zahtev, ki jih avtorja navajata, je v drugem delu tega poglavja (Stålhane & Hanssen, 2008).

Stålhane in Hanssen dopuščata možnost, da se vse zahteve ISO 9001 doda kot uporabniške zgodbe v agilni proces, vendar opozarjata, da obstaja zgornja meja, koliko dokumentacije se lahko zahteva od agilnega procesa, da ne trpi koncept agilnosti. Avtorja sta prepričana, da je mogoče združiti skladnost s standardom in agilni koncept tako, da kombiniranje prinese podjetju kar največje koristi. Poudarita, da mora podjetje dovolj prožno uporabljati izraze, kot so planiranje in dokaz o skladnosti, hkrati pa pazljivo izbirati takega revizorja, ki razume agilne metodologije razvoja programske opreme (Stålhane & Hanssen, 2008).

Namioka in Bran predlagata preprosto in uporabno rešitev združevanja agilnega razvoja in ISO standardov. Predlagata, da se vsaka interakcija (krajši časovni rok, do katerega dobi naročnik naslednjo različico aplikacije) izvede kot svoj projekt. Tako izgine večina težav, ko agilni razvoj težko ustreza ISO 9001, saj se te pojavijo večinoma le pri tistih zahtevah, ki se nanašajo na realizacijo proizvoda. Kot komentirata Stålhane in Hanssen, je takšna rešitev sicer uporabna, ni pa najbolj učinkovita, saj je treba vpeljati dodatne interakcije zgolj za pripravljanje dokumentacije (Namioka & Bran, 2004; Stålhane & Hanssen, 2008).

Boehm in Turner razpravljata in na različnih primerih prikažeta, kako vzpostaviti pravilno ravnovesje med agilnostjo in disciplino. Poudarita, da vsako uspešno podjetje v današnjem, hitro spreminjajočem se svetu, potrebuje oboje: primerno mero prilagodljivosti in discipline (Boehm & Turner, 2004).

McMichael in Lombardi opisujeta, kako so v podjetju Primavera Systems vzpostavili sistem vodenja kakovosti, ki ustreza ISO 9001 standardu in hkrati obdržali prednosti, ki jih

zagotavljata metodologiji Scrum in XP. V članku razložita, da naj bi uporaba tako Scrum, kot XP metodologije hkrati izpolnila vse zahteve ISO standarda. Poudarita tudi, da ISO 9001 ne enači kakovosti, pač pa skladnost s standardi zagotavlja, da podjetje res upošteva dobre prakse agilnega razvoja, ki si jih je predpisalo v sistemu vodenja kakovosti in s katerimi dosega kakovost svojih produktov (McMichael & Lombardi, 2007).

Vriens razloži, da večina je zahtev standarda ISO 9001 neodvisna od metodologije, s katero podjetje razvija programsko opremo. Na primeru navaja, kako je njihov majhen, vendar hitro rastoči oddelek Software Engineering Services v okviru Phillpis Research Organisation nadgradil uporabo XP na hkratno uporabo XP in Scrum metodologije in se registriral tako pri ISO 9001 kot pri CMM Level 2. Razlaga, da ob vpeljavi standarda ISO 9001 niso odkrili večjih neskladij, ki bi omejevala agilnost že vzpostavljenega procesa. Največji izziv je bilo primerno definirati nekatere zahteve ISO standarda in izbrati primernega presojevalca, ki bi bil pripravljen na zahteve ISO standarda pogledati z zornega kota agilnega razvoja. Podjetju sicer ni bilo treba bistveno prilagajati metodologije razvoja, moralo pa je uvesti nove meritve, da je lahko spremljalo učinkovitost in napredovanje sistema vodenja (Vriens, 2003).

Sam bi opozoril na olajševalno okoliščino pri združevanju standarda in metodologij primera, ki ga navaja Vriens, namreč na dejstvo, da niso delali izdelka za znanega naročnika, ampak t.i. produkt »of the shelf«, ki je namenjen prodaji neznanim kupcem.

Eden bolj kakovostnih člankov je Wrightov, v katerem opisuje uspešno certifikacijo ISO 9001 v podjetju, ki uporablja XP metodologijo razvoja. Avtor opisuje postopek certifikacije v podjetju Workshare, ki je metodologijo XP začelo uporabljati leta 2001, postopek pridobivanja certifikata pa je potekal v letih 2002 in 2003 ter je bilo tako med prvimi ISO certificiranimi XP podjetji na svetu. Ena od sprememb, ki so jih morali v podjetju narediti, je bila prenos table, na kateri se beleži status implementacije uporabniških zgodb v digitalizirano obliko. Čeprav avtorji s področja agilnih metodologij pogosto poudarjajo, da je tabla v skupnem prostoru ekipe primernejša, saj vse člane ekipe v vsakem trenutku opomni kakšen je status določenih nalog, pa digitalizirana oblika takšne table omogoča sledljivost, ki jo zahteva ISO 9001. Avtor priznava, da je podjetje uporabilo dokaj formaliziran proces, ki morda ni primeren za manjše XP ekipe, vendar ne vidi ovir, da ne bi pridobilo certifikata tudi podjetje z manj formaliziranim procesom (Wright, 2003).

Wright v članku poudari dve zanimivi trditvi. Pravi, da je vse, kar zahtevajo ISO standardi, revizija koristi XP metodologije za razvoj programske opreme in dokaz, da so XP postopki v podjetju tudi uporabljeni. Na tak način namreč podjetje, ki uporablja XP metodologijo, zagotovi, da upošteva model PDCA. Ključ do uspešne implementacije standarda, brez negativnih stranskih učinkov, v podjetju z XP razvojem programske opreme pa je uvedba takšnih postopkov dela, da postane priprava zahtevanih dokumentov sistema vodenja kakovosti po ISO 9001 naravni element procesa in ne element, narejen izključno zaradi skladnosti s standardom. Avtor priznava, da so se zaradi potrebe po skladnosti s standardom

pojavila dodatna dela, predvsem na področju priprave in posodabljanja različnih dokumentov, ki podrobno opisujejo že obstoječi delovni proces. Ker se avtor v članku poglablja le v področje dela, ki ga opravlja razvojna ekipa, ugotavlja, da standard ne povzroča dodatnega, obremenilnega dela. Ravno osredotočanje avtorja zgolj na proces razvoja je po mojem mnenju najšibkejša točka sicer kakovostnega članka, saj opisuje izpolnjevanje ISO 9001 z XP metodologijo le v točkah od 7.3.2 do 7.3.7. Kako so te točke standarda povezali z praksami XP metodologij, je opisano v drugem delu tega poglavja (Wright, 2003).

Victoria na dveh praktičnih primerih akademskih projektov, v katerih so uporabljali XP metodologijo, razišče, v kolikšni meri so izpolnjevali TickIT-jev vodnik za izpolnjevanje ISO 9001. Ker je pri obeh projektih šlo za neprofitno delo študentov, vse zahteve standarda niso bile relevantne, vendar sta po obsegu, zahtevani končni kakovosti in številu sodelujočih dober primer, ki bi ga lahko prenesli tudi v podjetje, ki deluje na trgu. Projektni podjetji, ki ju raziskuje avtor, sta z uporabo dobrih praks XP metodologije izpolnili 20 % zahtev ISO 9001, 24 % sta jih delno izpolnili, 33 % zahtev sploh nista izpolnili, 23 % zahtev pa ni bilo relevantnih, ker sta ekipi delali na študentskih, neprofitnih projektih (Victoria, 2004).

Erharuyi na temelju raziskave med petimi nigerijskimi podjetji ugotavlja, koliko so ta podjetja z XP razvojem programske opreme izpolnjevala ISO standard 9001 v točkah 7 in 8. Sklepa, da bi bile v točkah, kjer podjetja niso izpolnjevala standarda, potrebne le manjše prilagoditve načina dela. Nekatere ugotovitve avtorja po posameznih točkah so povzete v drugem delu tega poglavja (Erharuyi, 2007).

Drugače od večine avtorjev se Nawrocki s soavtorji ne sprašuje, kako certificirati XP podjetje z ISO 9001 standardom, temveč, kako v podjetje, ki že ima pridobljen certifikat, vpeljati agilno metodologijo razvoja. Predlaga, da se podjetja takšnega projekta lotijo s t.i. XP Maturity Model (XPMM), ki vpeljuje XP metodologijo po kosih, stopnjo za stopnjo. Štiri stopnje modela XPMM so:

- Stopnja 1 – Odnosi z naročnikom (angl. *Customer Relationship Management*)
- Stopnja 2 – Zagotavljanje kakovosti (angl. *Product Quality Assurance*)
- Stopnja 3 – Programiranje v parih (and. *Pair Programming*)
- Stopnja 4 – Izvrševanje projekta (angl. *Project performance*)

Avtor vsako stopnjo razčleni na dobre prakse XP, ki izpolnjujejo posamezne zahteve standarda, navede primere, ko mora podjetje spremeniti način dela, da je še skladno s standardom, ter izpostavi potrebne spremembe pri interpretaciji standarda. Ta članek je dobra osnova za vse, ki v projektih že delujejo po ISO 9001, vendar bi v svoje delovne procese radi vnesli več agilnosti (Nawrocki, Jasiński, Walter & Wojciechowsk, 2002).

Iz raziskav na temo uvedbe ISO 9001 v podjetje z agilnim razvojem programske opreme lahko izluščimo nekatere njihove skupne značilnosti. Ne glede na to, ali delajo avtorji

raziskave na temelju opazovanj podjetij v praksi ali proučujejo teorijo, ugotavljajo, da agilni razvoj vpliva predvsem na način izpolnjevanja zahtev, ki se nanašajo na realizacijo proizvoda, se pravi na zahteve iz sklopa 7 (realizacija proizvoda) in na zahteve iz sklopa 8 (merjenje, analize in izboljševanje), saj le zahteve omenjenih sklopov vodijo do potrebe po prilagoditvi določenih postopkov dela. Pri zahtevah iz sklopov 4 (sistem vodenja kakovosti), 5 (odgovornost vodstva) in 6 (vodenje virov) avtorji zapletov večinoma ne navajajo. Večina zahtev standarda je izpolnjenih z doslednim spoštovanjem praks agilnega razvoja, nekaj praks je treba prilagoditi, pri nekaterih pa lahko v določenih primerih pride do večjih zapletov.

Za boljši pregled v naslednjem poglavju predstavljam zahteve standarda in pregled, katere zahteve dosledna uporaba določene agilne metodologije avtomatsko izpolnjuje, pri katerih je potrebno prilagoditi način dela in katere od zahtev nimajo vsebinske zveze z izbiro metodologije razvoja programske opreme.

4.2 Skladnost zahtev standarda ISO 9001 z uporabo agilnih metodologij

V poglavju 4.1 je predstavljen pregled raziskav različnih avtorjev na temo hkratne uporabe standarda ISO 9001 in agilnih metodologij. V tem poglavju pa so predstavljene vse točke standarda in, kjer je potrebno, tudi smernice standarda ISO 90003 (Internal Organization for Standardization, 2004). Pod vsako zahtevo ali sklopom zahtev navajam tudi komentarje nekaterih, prej predstavljenih avtorjev o težavah, ki se lahko pojavijo ob uvajanju ISO 9001 v podjetje z agilnim razvojem programske opreme. Največkrat omenjam mnenja avtorjev Stålhana in Hanssna, Wrighta, Victorie, Erharuyija in Nawrockega. Kjer se sklepi avtorjev glede posameznih zahtev ne ujemajo, podajam na koncu zahteve ali sklopa zahtev, še svoje mnenje.

Standard ISO 9001 je razčlenjen na osem poglavij. Prvo poglavje je splošen opis standarda in njegove uporabe, drugo opisuje povezavo ISO 9001 z drugimi ISO standardi, tretje pa razlaga nekatere izraze in definicije, uporabljene v standardu. Prva tri poglavja torej ne vsebujejo zahtev za implementacijo, zato začenjam podroben opis s četrtim poglavjem.

4.2.1 (4) Sistem vodenja kakovosti

4.2.1.1 (4.1) Splošne zahteve

Podjetje mora razviti in uvesti sistem kakovosti. Za to mora identificirati procese, njihovo povezanost in medsebojne vplive. Tem procesom mora določiti merila in metode, s katerimi jih učinkovito obvladuje. Zagotoviti mora potrebne informacije za podporo delovanja in za nadzor procesov. Nenehno se mora izboljševati s pomočjo meritev, spremljanja in analiziranja.

Ta točka standarda nima posebnih zahtev, zato je ni treba obravnavati posebej: podjetje namreč izpolnjuje zahteve te točke, če izpolnjuje vse zahteve v sledečih točkah.

4.2.1.2 (4.2) Zahteve glede dokumentacije

Točka 4.2.1. Splošne zahteve navaja, katera poglavja so podlaga za dokumentacijo. Ta točka nima nobene specifične zahteve. Podjetje mora opredeliti, kateri so njeni procesi, kako ti medsebojno delujejo, kateri viri so potrebni, da nastane izdelek, in kako bo procese merilo in izboljševalo. Skupaj s poslovnikom kakovosti (zahteva točke 4.2.2) mora vzpostaviti še sistem za obvladovanje dokumentacije (zahteva točke 4.2.3) in obvladovanje zapisov (zahteva točke 4.2.4).

Kot zahtevo, ki jo je teže izpolniti, navajata Stålhane in Hanssen (Stålhane & Hanssen, 2008) podtočko 4.2.1.d, ki zahteva, da sistem vodenja kakovosti vsebuje dokumente, ki so potrebni za učinkovito planiranje, realizacijo in nadzor procesa. Kot rešitev avtorja predlagata opis metodologije, ki jo podjetje uporablja pri razvoju, in zapis podrobnosti, na primer dolžino iteracije in način, kako se sledi uporabniškim zgodbam in se jih potrjuje.

Kar zadeva zahteve glede dokumentacije, zahteva točka 4.2.4 (Obvladovanje zapisov) največ truda za učinkovito sinhronizacijo agilnih metodologij in ISO 9001. Stålhane in Hanssen (2008) navajata, da je treba med posameznima iteracijama ob predstavitvi implementiranih funkcionalnosti (pri Scrumu je to pregled iteracije ali angl. *Sprint Review*), zabeležiti naročnikovo potrdilo o veljavnosti implementiranega in druge naročnikove komentarje.

Primera, ki ju raziskuje Victoria (2004), sta z upoštevanjem agilnih metodologij že izpolnjevala zahteve po izvedbi poslovnika kakovosti (točka 4.2.2), zahteva po obvladovanju dokumentov (točka 4.2.3) pa je bila le delno izpolnjena, saj XP ne zahtevajo podrobne specifikacije naročenih zahtevkov in se v veliki meri zanašajo na sočasno sodelovanje predstavnika odjemalca. Primera zahteve po obvladovanju zapisov (točka 4.2.4) nista izpolnila saj sestankov niso snemali, tako kot tudi niso shranili rezultatov testiranj, saj tega metodologija XP ne zahteva.

Čeprav je zahteva po obvladovanju dokumentacije na videz ena težjih ob uvajanju standarda ISO 9001 v podjetje z agilnimi metodologijami razvoja programske opreme, ugotovimo, da se da večino zahtev izpolniti brez večjega vpliva na agilnost razvoja. Poleg tega da mora podjetje popisati lastnosti metodologije, ki jo uporablja pri razvoju, mora poskrbeti, da po vsakem zaključku iteracije prejme pisno naročnikovo potrdilo, da je dobil tisto, kar so se dogovorili na začetku iteracije, z njegovimi komentarji in pripombami vred. Opazovanje primerov podjetij, ki uporabljata metodologijo XP, nakazuje še več drugih neskladij, ki pa ne povzročajo veliko dodatnega dela in ne onemogočajo agilnosti razvoja. V podjetju morajo na primer poskrbeti, da so sestanki posneti ali zapisani njihovi povzetki, uveljaviti morajo sistem sledenja rezultatom testov in vzpostaviti sistem sledenja posameznim uporabniškim zgodbam.

4.2.2 (5) Odgovornost vodstva

Najvišje vodstvo v podjetju se mora dobro zavedati tega pomembnega dela standarda. Vodstvo je namreč odgovorno za določanje politike in ciljev ter za pregled sistemov vodenja kakovosti, hkrati pa tudi za obveščanje o učinkovitosti sistema znotraj podjetja.

4.2.2.1 (5.1) Zavezanost vodstva

Najvišje vodstvo v podjetju mora podpirati razvoj, izvajanje in stalno izboljševanje učinkovitosti sistema vodenja kakovosti. Takšno delovanje mora tudi dokazati.

Zavezanost vodstva pri agilnem razvoju programske opreme ni vprašljiva, zato ta zahteva ne ovira agilnosti razvoja programske opreme, kar potrjujeta tudi primera, ki ju navaja Victoria (2004).

4.2.2.2 (5.2) Osredotočenost na odjemalce

Vodstvo je odgovorno za naročnikovo zadovoljstvo, ki ga doseže s prepoznavanjem naročnikovih potreb in z zadovoljitvijo le-teh. Zagotoviti mora, da podjetje naročnikovo zadovoljstvo spremlja in meri (nanaša se na podtočki 7.2.1 in 8.2.1).

Osredotočenost na odjemalce je ena od glavnih vrednot agilnih metodologij. Njihova značilnost je pogosto sodelovanje z odjemalcem tudi med posameznimi iteracijami. Vrhovno vodstvo dokaže skladnost s to zahtevo že s tem, ko v podjetju vzpostavi razvoj s katero od agilnih metodologij, in poskrbi, da jo v podjetju dosledno izvajajo.

4.2.2.3 (5.3) Politika kakovosti

Vodstvo mora zagotoviti, da sistem vodenja kakovosti v podjetju dosega svoj namen in cilj. Zagotavljati mora tudi, da so zahteve sistema vodenja kakovosti izpolnjene in da se učinkovitost sistema neprestano izboljšuje. Vodstvo je tudi odgovorno za primerno razlago sistema vodenja vsem, ki vplivajo na njegovo izvedbo in za to, da ga tudi razumejo.

Stålhane in Hanssen (2008) se osredotočita na podtočki 5.3. a: »Vodstvo mora zagotoviti, da je politika kakovosti primerna za cilj podjetja« in 5.3. b: »Vodstvo mora zagotoviti, da vzpostavljena politika kakovosti vsebuje zavezo za izpolnjevanje zahtevkov in neprestano napredovanje učinkovitosti in kakovosti dela«. Avtorja navajata, da dobro vzpostavljen agilni proces, ki mu vodstvo zagotovi vse potrebne pogoje, sledi cilju podjetja: »redni dobavi uporabne programske opreme v optimalnem razmerju med ceno in hitrostjo dobave«. Dobro delujoč agilni proces zagotavlja zavezanost vseh sodelujočih za dobavo zahtevanih funkcionalnosti, saj se morajo ob začetku vsake iteracije vsi sodelujoči pri razvoju strinjati, da so sposobni izvesti in sprejmejo planirano v naslednji iteraciji. Tudi Victoria (2004) navaja,

da so politiko kakovosti v obeh opazovanih projektih razvili vodje projektov, razvijalci pa so jo dobro razumeli.

Če povzamem: ne le da zahteve iz te točke standarda nikakor ne otežujejo agilnega razvoja programske opreme, temveč je skladnost s to točko standarda nujen pogoj za dobro delujočo vzpostavitev katerekoli od agilnih metodologij.

4.2.2.4 (5.4) Planiranje

Vrhovno vodstvo podjetja mora predstaviti merljive cilje kakovosti (točka 5.4.1), ki so v skladu s politiko kakovosti in z zahtevami za proizvod. Ko vodstvo določi cilje, mora poskrbeti za ustanovitev, dokumentiranje, razvoj, vzdrževanje in neprestano izboljševanje sistema vodenja kakovosti (točka 5.4.2). Cilji, ki jih določi vodstvo, morajo izpolniti SMART kriterije – določen (angl. *Specific*), merljiv (angl. *Measurable*), sprejemljiv (angl. *Acceptable*), realističen (angl. *Realistic*) in časovno omejen (angl. *Time-bund*).

Kot učinkovit način, da podjetje z agilnim razvojem programske opreme izpolnjuje zahteve točke 5.4.1, Stålhane in Hanssen (2008) predlagata uporabo agilne metodologije Evo. Ta metodologija poudarja merljive cilje kakovosti in jo je mogoče preprosto vključiti v proces, ne glede na to, po kateri agilni metodologiji se ravna podjetje. Značilnosti metodologije Evo podrobno opiše Gilb (Gilb, 2005).

Opazovana projekta, ki ju raziskuje Victoria (2004), nista izpolnjevala zahtev po definiranju produktov razvoja programske opreme, ki jih navaja ISO 90003 pod točko 5.4.2. Neskladnosti so se pojavile pri planiranju priprave dokumentov o naročnikovih zahtevkih in pri natančnem načrtovanju. Projekta sta uporabljala metodologijo XP, ki nima posebnih zahtev glede zapisov nefunkcionalnih zahtevkov (hitrost, učinkovitost, varnost, robustnost…), saj naj bi lastnosti metodologije same zagotavljale izpolnitev teh zahtev, ne da bi jih posebej dokumentirali.

Planiranje se lahko nanaša tako na raven podjetja, kot na raven posameznega projekta. Čeprav se zahteve točke 5.4.2 nanašajo le na planiranje na ravni podjetja (zahteve planiranja na ravni projekta so navedene v točki 7.1), ISO 90003 navaja (5.4.2 a), da mora podjetje zagotoviti uporabo primernih metodologij razvoja programske opreme za posamezne vrste projektov. Ne glede na to, katero agilno metodologijo uporablja podjetje, jo je priporočljivo kombinirati z Evo, ki izpolni zahtevo po merljivosti.

Eden večjih izzivov za podjetje pri uvajanju ISO 9001 je planiranje definiranja naročnikovih zahtevkov in podrobnega načrtovanja, ki ju navaja ISO 90003 v točki 5.4.2 b. Za uspešno načrtovanje teh produktov pri razvoju programske opreme mora podjetje najti pravo ravnovesje med agilnostjo in disciplino (Boehm & Turner, 2004).

4.2.2.5 (5.5) Odgovornost, pooblastila in komuniciranje

Vodstvo mora zagotoviti natančno definicijo pooblastil in odgovornosti glede posameznih nalog (točka 5.5.1). Enega od zaposlenih mora pooblastiti za izvedbo sistema kakovosti – to je t.i. predstavnika vodstva za sistem vodenja kakovosti (točka 5.5.2). Zagotoviti mora tudi postopke komuniciranja in izvajanje le teh (točka 5.5.3).

Pogled agilnih metodologij na določanje odgovornosti se precej razlikuje od tradicionalnih metodologij, saj agilne metodologije ne nalagajo odgovornosti za izvršitev projekta eni osebi temveč celotni ekipi. Ker mora ob vsaki začeti iteraciji vsak član ekipe prevzeti odgovornost, da bo planirano v naslednji iteraciji tudi narejeno, ni težav pri izpolnjevanju zahtev točke 5.5.1.

Čeprav v nobeni raziskavi o tej tematiki točka 5.5.2 ni omenjena kot težko izpolnjiva, se strogo gledano lahko zatakne pri izpolnitvi njenih zahtev. Agilne metodologije namreč predvidevajo eno osebo, ki poskrbi, da proces nemoteno teče (pri Scrum-u je to Scrum Master), ni pa ta oseba odgovorna za kakovost in komunikacijo z naročnikom. Za slednji so pooblaščeni in odgovorni vsi člani ekipe.

Dobra komunikacija, ki jo zahteva točka 5.5.3, tako notranja kot zunanja, je ena značilnosti prve vrednote agilnih metodologij, zato vse agilne metodologije skrbijo za dobre procese notranje komunikacije.

4.2.2.6 (5.6) Vodstveni pregled

Točka 5.6.1 navaja, da je vodstvo odgovorno za izvajanje vodstvenih pregledov v načrtovanih presledkih. V teh pregledih mora vodstvo ugotoviti, ali je politika kakovosti, po kateri se ravna podjetje, primerna in učinkovita in kakšni so možni ukrepi za napredovanje. Za učinkovito izvajanje teh pregledov mora pridobiti ustrezne vhodne podatke, ki jih zahteva točka 5.6.2. Rezultate pregleda in morebitne ukrepe mora vodstvo v podjetju predstaviti, da zagotovi povratne informacije s strani vodstva, kakor jih zahteva točka 5.6.3.

Stålhane in Hanssen (2008) posebej navajata podtočke 5.6.2 a (rezultati presoj), 5.6.2 b (povratne informacije odjemalcev) in 5.6.2 c (učinkovitost procesov in skladnost proizvodov). Obravnavo tematike teh točk predvidijo tudi agilne metodologije ob koncu posamezne iteracije (sprint review), vendar ne zahtevajo dokumentiranja ugotovljenega. Avtorja navajata, da mora podjetje za skladnost s standardi ISO 9001 voditi zapisnike sestankov ob koncu vsake iteracije ter tako zagotoviti ustrezne vhodne podatke za izvedbo vodstvenega pregleda.

Poudaril bi, da se zahteve iz točke 5 ne nanašajo na posamezen projekt, ampak na celotno podjetje. Navodila, ki jih navajata Stålhane in Hanssen, je tako treba razumeti kot navodila za dopolnitev sistema vodenja kakovosti. Ko podjetje planira uporabo določene metodologije za

določene vrste projektov, kot zahteva ISO 90003 v točki 5.4.2 a, mora to metodologijo dopolniti tako, da bodo izdelani tudi ustrezni produkti, ki bodo kasneje vhodni podatki za vodstveni pregled. Za skladnost s preostalimi zahtevami točke 5.6 ni bistveno, katero metodologijo razvoja programske opreme uporablja podjetje.

4.2.3 (6) Vodenje virov

Novi standard bolj poudarja vire, ki jih mora podjetje zagotoviti, da bo naročnik dobil tisto, o čemer se je dogovoril. To niso le človeški, temveč tudi fizični viri, na primer oprema, prostori in vse potrebne pomožne storitve. V podjetju morajo ugotoviti in zagotoviti vire, ki so potrebni za izvedbo sistema vodenja kakovosti ter za izpolnitev naročnikovih zahtevkov. S tem zagotovijo tudi izpolnitev zahtev točke 6.1 (Preskrba virov).

Točka 6.2 navaja zahteve v zvezi s človeškimi viri. Podjetje mora zagotoviti, da so človeški viri, ki lahko posredno ali neposredno vplivajo na zmožnost izpolniti zahteve izdelka (poleg zaposlenih so to lahko še zunanji sodelavci), ustrezni in imajo vsa potrebna znanja in sposobnosti. Za uresničitev te zahteve mora podjetje zagotoviti izobraževanja, treninge in druge postopke izboljševanja sposobnosti zaposlenih. Zaposleni in drugi sodelujoči pri razvoju produkta se morajo zavedati, kako in s čim prispevajo k procesu izvajanja sistema vodenja kakovosti.

Točka 6.3 navaja zahteve v zvezi z infrastrukturo, točka 6.4 pa v zvezi z delovnimi razmerami. Podjetje mora ugotoviti in zagotoviti primerno infrastrukturo in delovne razmere, da bo produkt zadovoljil naročnikove zahteve. Potrebno infrastrukturo je treba sproti vzdrževati in po potrebi dopolnjevati.

Pri razvoju z agilnimi metodologijami je celotna ekipa odgovorna za kakovostno rešitev in zadovoljitev želja naročnika. Vsak razvijalec je odgovoren tudi za testiranje. Tudi za prenos znanja med razvijalci imajo vse agilne metodologije vzpostavljene različne tehnike. Točko standarda, ki se nanaša na človeške vire, tako izpolnjuje vsako podjetje, ki je uspešno vpeljalo katero od agilnih metodologij. Zahteve iz te točke sta izpolnjevala tudi primera, ki ju raziskuje Victoria (2004) in v katerih uporabljajo metodologijo XP.

Nawrocki et al. (2002), ki opisuje prehod s tradicionalne metodologije razvoja programske opreme h kateri od agilnih metodologij v podjetju, ki je že skladno z zahtevami certifikata ISO 9001, poudarja pomembnost izpolnjevanja zahtev glede infrastrukture in delovnega okolja. Potrebe agilnega razvoja se namreč razlikujejo od potreb razvoja z tradicionalnimi metodologijami. Največja razlika pri zahtevani infrastrukturi je prostor. Agilne metodologije namreč zahtevajo, da vsi razvijalci delajo v istem prostoru, če je le mogoče. Bolj zapleteno je izpolniti primerne delovne razmere za agilni razvoj. Primera takšne zahteve, ki jih Nawrocki ne navaja podrobno, sta sprememba organizacijske strukture podjetja, ki ne sme biti preveč

hierarhična, na ravni posameznega projekta pa mora biti skoraj popolnoma ploščata, in politika nagrajevanja, ki mora spodbujati timsko delo.

4.2.4 (7) Realizacija proizvoda

Ta del standarda obsega največ zahtev in pokriva vse vidike nastajanja produkta. Ker se izbira metodologije razvoja programske opreme nanaša predvsem na realizacijo proizvoda, najbolj vpliva na izpolnjevanje zahtev te točke, čeprav se v več primerih nanaša tudi na izpolnjevanje zahtev drugih točk.

4.2.4.1 (7.1) Planiranje realizacije proizvoda

Proces realizacije proizvoda mora biti planiran in implementiran skladno z zahtevami preostalih procesov sistema vodenja kakovosti. Procesi morajo biti razviti tako, da produkta ni mogoče razvijati, ne da bi jih upoštevali. ISO 90003 razčleni točko planiranja realizacije proizvoda na izbiro ustrezne metodologije razvoja programske opreme za posamezen projekt in na planiranje zagotavljanja kakovosti.

Stålhane in Hanssen (2008) se osredotočita na podtočki ISO 9001 7.1. a in 7.1. b, ki ju podjetje izpolni, če podrobno opiše agilno metodologijo, ki jo bo uporabljalo pri posameznem projektu (lastnosti metodologije, proces planiranja, trajanje iteracije, načini dokumentiranja uporabniških zgodb…).

Victoria (2004) navaja, da sta primera, ki ju raziskuje, le delno izpolnjevala to točko standarda. Agilne metodologije namreč predvidevajo podrobno planiranje le za eno do dve iteraciji vnaprej, dolgoročni plani pa so bolj ohlapni. Ravno tako na projektu niso zapisovali načina planiranja za kasnejše preverjanje.

Poudaril bi, da ISO 90003 posebej navajajo, da ISO 9001 ne navija za uporabo določene metodologije razvoja programske opreme. Planiranje le za eno do dve iteraciji v tej točki ne pomeni kršitve zahtev standarda, saj ta točka zahteva, da za projekt natančno določijo in opišejo postopke in navodila, ki zagotavljajo upoštevanje sistema vodenja kakovosti, za specifičen projekt. Sistem vodenja kakovosti, ki je narejen za specifičen projekt mora biti skladen s sistemom vodenja kakovosti, ki velja za podjetje in lahko po potrebi vključuje njegove dele.

4.2.4.2 (7.2) Postopki povezani z naročnikom

Kakor zahteva podtočka 7.2.1, mora podjetje pred podpisom pogodbe ugotoviti vse zahteve, v zvezi s konkretnim naročnikom. To so zahteve na področju razvoja posameznega izdelka, dobave, po-dobavne storitve, kakor tudi zahteve, ki jih naročnik ni posebej navedel, vendar

jih je treba izpolniti za normalno uporabo. Takšne zahteve so lahko nefunkcionalne narave, na primer hitrost delovanja, zakonske, regulatorne ter druge zahteve.

Točka 7.2.2 navaja zahteve po pregledu naročnikovih zahtev, povezanih s produktom. Pred podpisom pogodbe z naročnikom morajo v podjetju preveriti, ali so vse zahteve dovolj natančno definirane, nedvoumne in razumljive obema poslovnima partnerjema. V podjetju se morajo tudi prepričati, da lahko izdelajo produkt, ki vse zahteve iz pogodbe izpolni zadovoljive kakovosti in v roku.

Zelo pomembna točka, še posebno pri razvoju programske opreme, je točka 7.2.3: komunikacija z naročnikom. Podjetje mora ugotoviti in vzpostaviti komunikacijske tokove z naročnikom, ki so učinkoviti in obsegajo komunikacijo o lastnostih produkta, naročnikovih povpraševanjih, pogodbi z vsemi prilogami, naročnikovih komentarjih in pripombah. Naročnik mora biti obveščen o stanju razvoja in nadgradnjah produkta.

Za izpolnitev podtočke 7.2.1. a, ki zahteva določanje lastnosti produkta, podanih s strani odjemalca, vključno z zahtevami dobave (inštalacije) in po-dobavnih aktivnostih produkta, Stålhane in Hanssen (2008) svetujeta, da podjetje v plan iteracij doda dve ali več iteracij na koncu projekta. Dodane iteracije izpolnjujejo zahteve dobave in po-dobavne zahteve, ki jih agilne metodologije ne omenjajo.

Za ohranitev agilnosti razvoja programske opreme in hkrati izpolnjevanje zahtev standarda ISO 9001 je najbolj problematična točka 7.2.2, če jo je treba upoštevati zelo striktno. Ta točka zahteva, da podjetje podrobno pregleda naročnikove zahtevke, preden se zaveže za dobavo produkta. Stålhane in Hanssen (2008) poudarita podtočki 7.2.2. a (podjetje mora preveriti, da so naročnikovi zahtevki definirani) in 7.2.2. c (podjetje mora preveriti, da je sposobno zahtevano dobaviti). Strogo upoštevanje teh zahtev bi pomenilo, da so vse lastnosti bodočega produkta podjetju znane že pred podpisom pogodbe.

Agilne metodologije sicer zberejo vse zahtevane funkcionalnosti produkta vnaprej, vendar se nabor funkcionalnosti lahko med potekom projekta še spremeni, kakor tudi funkcionalnosti niso podrobno specificirane, dokler posamezna funkcionalnost ni na vrsti za izvedbo v naslednji iteraciji. Tak način dela pri agilnih metodologijah temelji na predpostavki, da vseh podrobnosti izdelka ni mogoče dognati pred pričetkom razvoja, da pa je treba dokumentirati najbolj pomembne lastnosti produkta. Pri Scrumu se nabor lastnosti produkta (uporabniških zgodb ali User stories) imenuje Product Backlog. Ta zbirka vseh uporabniških zgodb mora biti vzpostavljena pred prvo iteracijo in jo je treba med samim razvojem redno posodabljati z novimi informacijami. Tak način dela daje naročniku več svobode in povečuje verjetnost, da bo dobil tak izdelek, kakršnega potrebuje. Stålhane in Hanssen (Stålhane in Hanssen, 2008) zaključita, da je pri tej točki veliko odvisno od izbire presojevalca, ki lahko način skladnosti s to točko, kakršnega ponujajo agilne metodologije, sprejme ali pa tudi ne. V primeru da presojevalec takšnega izbora lastnosti produkta ne sprejme kot skladnega z ISO 9001, razvoj z

agilnimi metodologijami ne more biti skladen s standardom, saj bi podrobno načrtovanje vseh lastnosti produkta uničilo koncept agilnega razvoja.

Primera, ki ju raziskuje Victoria (2004), nista izpolnjevala več zahtev iz točk 7.2.1 in 7.2.2. Zahteve naročnika (uporabniške zgodbe) so dokumentirali zelo splošno in jih niso dopolnili, ko so bile znane podrobnosti, kar je onemogočalo sledljivost določeni zahtevi. Ravno tako niso dokumentirali predvidene dobavne in po-dobavnih aktivnosti. Tako lahko na temelju raziskovanih primerov vidimo, da zgolj dosleden razvoj z določeno agilno metodologijo ne zadostuje za izvajanje projekta v skladu z zahtevami točke 7.2. Najnujnejša je zagotovitev sledljivosti določeni zahtevi naročnika in dosledno posodabljanje uporabniških zgodb, ko so znane dodatne podrobnosti o zahtevah. Nobena agilna metodologija ne zahteva sledljivosti, čeprav je ni težko zagotoviti, če se podjetje odloči za uporabo katerega od orodij za podporo posamezni agilni metodologiji. S takšnim orodjem je za vsako zahtevo mogoče izvedeti, kdo se je kdaj ukvarjal z njo, dopolnjeval njene specifikacije, jo testiral, katera koda je bila spreminjana zaradi te zahteve in kakšen je njen status. Primeri takšnih orodij so Jira, Bugzilla, ScrumWorks in mnogo drugih.

Za uspešen agilni razvoj in tekoče delo razvijalcev opravlja zelo pomembno vlogo predstavnik naročnika (angl. *customer representative*). Njegova naloga je pregledati uporabniške zgodbe, ki so predvidene za izvedbo v naslednji iteraciji in sproti komentirati uporabnost določenih delov programskega paketa. Nawrocki et al. (2002) poudarja, da je lahko dobro definiranje njegove vloge (kar zahteva točka 7.2.2.3 v ISO 90003), ključno za učinkovit agilni razvoj. Opozoril bi, da zahteva točke 7.2.2.3 določa, da mora biti predstavnik naročnika samo ena oseba, kar v določenih primerih zahteva prilagoditev načina dela. Pri agilnem razvoju razvijalci namreč pogosto komunicirajo z različnimi predstavniki naročnika, odvisno od tematike. Za izpolnitev te zahteve je komunikacija z različnimi predstavniki sicer dovoljena, vendar mora predstavnik naročnika vedno potrditi vsak nov dogovor.

4.2.4.3 (7.3) Snovanje in razvoj proizvoda

Plan snovanja in razvijanja produkta, o katerem govori podtočka 7.3.1, mora biti skladen s sistemom vodenja kakovosti in s splošnim planom procesa realizacije produkta, o katerem govori točka 7.1. Vhodni podatki (specifikacije) za načrtovanje proizvoda, ki jih določa podtočka 7.3.2, morajo biti natančno določeni, njihove spremembe pa zabeležene. Vir vhodnih podatkov so lahko zahteve, pridobljene neposredno od naročnika (vir točka 7.2.1), lahko pa so izpeljane iz teh zahtev, lahko so tehnične omejitve ali pa so zakonske narave.

Rezultati faz snovanja in razvijanja, ki jih obravnava podtočka 7.3.3, morajo biti v takšni obliki, da jih je mogoče primerjati s pričakovanimi rezultati posamezne faze in da je mogoče ugotoviti, ali se ujemajo ali ne. Vsak rezultat faz snovanja in razvijanja je treba v vsaki fazi razvoja preveriti, ali je primeren in učinkovit za predvideno uporabo (zahteva točke 7.3.4) in verificirano ustreza predvidenemu iz prejšnje faze razvoja (zahteva točka 7.3.5). Ravno tako

mora za vsak izdelek obstajati dokazilo, da ustreza predvideni uporabi in da bo zadovoljil naročnika. Najbolj neposredno je dokazilo takrat, kadar lahko prikažemo delovanje v enakih razmerah, v kakršnih bo izdelek uporabljal naročnik (točka 7.3.6).

Pomemben del, še posebej pri razvoju programske opreme, so tudi spremembe med načrtovanjem in razvojem produkta. Standard v točki 7.3.7 zahteva, da so spremembe ustrezno dokumentirane in da prestanejo enaka preverjanja, kakršna prestane izvajanje začetnih naročnikovih zahtev. Realizacijo te točke podrobneje opisuje točka 7.5.3.

V sklopu planiranja snovanja in razvijanja produkta se Stålhane in Hanssen (2008) poglobita v podtočko 7.3.1 a, ki pravi, da mora podjetje določiti stopnje v snovanja in razvoja produkta. Agilne metodologije se ukvarjajo s planiranjem in kontroliranjem načrtovanja in razvoja produkta, čeprav se izvedba razlikuje od tradicionalnih metodologij, saj se fazi načrtovanja in razvoja izmenjujeta in ponavljata. Ker plan implementacije posamezne uporabniške zgodbe navadno snujejo skupinsko, na tablo, Stålhane in Hanssen opozorita, da je treba za izpolnjevanje zahteve te točke končni dogovor dosledno slikati in shraniti.

Victoria (2004) navaja, da primera, ki ju je proučeval, nista bila oziroma sta bila le delno skladna s točko 7.3.1. Načrtovanje je bilo planirano zelo splošno, saj XP predvidevajo spremembe kode (refactoring) vsakič, ko lahko to prispeva k preprostejši, učinkovitejši ali bolj berljivi izvedbi iste funkcionalnosti. Takšni popravki tudi niso bili dokumentirani. Tu ocenjujem, da agilnost razvoja ne bi trpela, če bi tudi vsako spremembo kode, ki ni namenjena novi funkcionalnosti, vodili kot svojo uporabniško zgodbo, z vso pripadajočo dokumentacijo.

Pred vsako iteracijo je narejen pregled uporabniških zgodb, ki jih bodo izvedli v sledeči iteraciji. Za te uporabniške zgodbe je treba z naročnikom zbrati vse potrebne informacije, tako funkcionalnih zahtev, kot zmogljivostnih. Kot navajata Stålhane in Hanssen (2008), agilne metodologije na tak način poskrbijo za skladnost procesa z zahtevami glede vhodnih podatkov. Teoretičnega odgovora, ki ga ponujata Stålhane in Hanssen, pa ne potrjujeta primera, ki ju opisuje Victoria (2004). S to točko se ne sklada noben od obravnavanih primerov, saj opis uporabniških zgodb ni zajel nefunkcionalnih zahtev (XP jih eksplicitno ne zahteva). Erharuyi navaja, da so bila podjetja, ki jih je obravnaval, skladna s to točko, saj so poleg dokumentov o uporabniških zgodbah uporabljala tudi t.i sprejemne teste (acceptance test). Ti testi so pogoj, da naročnik potrdi veljavnost določene funkcionalnosti. Zanje je značilno, da preizkusijo tudi nefunkcionalne zahteve in so načrtovani v sodelovanju z naročnikom. Podobno razlagajo tudi Nawrocki et al. (2002), ki za to, da bi dosegli skladnost s standardom, predstavnika naročnika poleg sodelovanja pri načrtovanju sprejemnih testov zadolži tudi za dokumentiranje vsebine teh testov. V praksi se pri tem pogosto zalomi, po čemer lahko sklepamo, da teorija in praksa ne pripeljeta do enakih zaključkov, kakor kažeta zgornja primera. XP sicer predvidevajo sprejemne teste, ki upoštevajo tudi nefunkcionalne zahteve, vendar ne zahtevajo natančnega popisa teh zahtev na začetku iteracije. Menim, da mora za skladnost z zahtevami točke 7.3.2 podjetje zagotoviti, da navede tudi nefunkcionalne

zahteve ali med lastnostmi funkcionalnih uporabniških zgodb ali pa kot ločene uporabniške zgodbe.

Za skladnost z zahtevami točke 7.3.3, Stålhane in Hanssen (2008) navajata, da tudi pri uporabi metodologije XP ni ovir za vključitev okvirnega načrtovanja v prvo iteracijo, v podrobnega načrtovanja vsake posamezne kasnejše ob njenem začetku. Rezultati načrtovanja in razvoja morajo biti podani v takšni obliki, da jih je mogoče preveriti glede na vhodne podatke v posamezno fazo in potrditi pred dobavo. Primera, ki ju raziskuje Victoria (2004), sta le delno izpolnjevala to točko. Specifikacije so namreč pogosto popisovale delovanje funkcionalnosti potem, ko je bila ta že narejena. Specifikacije, narejene na tak način, sicer lahko služijo za kasnejše lažje vzdrževanje kode, ne izpolnjujejo pa zahteve te točke, da naj bo izhode posamezne točke mogoče primerjati z vhodi. Erharuyi (2007) vidi manj ovir za skladnost s to točko, saj kot primer skladnosti navaja, da so izhodi načrtovanja in razvoja v obliki uporabne, delujoče kode, ki jo takoj testirajo, ko jo vključijo v projekt.

Čeprav je na prvi pogled morda nelogično, da imajo raziskovalci podjetij z agilnim razvojem tako različne poglede na skladnost z zahtevami točke 7.3.3, je treba poudariti, da se agilne metodologije med seboj razlikujejo in nimajo vse predpisanih enakih aktivnosti. Menim, da mora podjetje, ne glede uporabljeno metodologijo, v prvi iteraciji opraviti vsaj okvirno načrtovanje, glede podrobnega načrtovanja pa se mora odločiti za obliko. Lahko uporablja testno voden razvoj (angl. *Test-Driven Development*) ali pa pred implementacijo posamezne uporabniške zgodbe vselej izdelati podrobne specifikacije zanjo. Bistveno je, da je pred izvedbo jasno kakšen bo rezultat, in da je to mogoče dokazati.

Pregled načrtovanja in razvoja je treba sistematično izvajati v ustreznih stopnjah razvoja. Stålhane in Hanssen (2008) se poglobita v podtočko 7.3.4. a, ki zahteva, ovrednotenje uporabnosti rezultatov načrtovanja in razvoja, da bodo ustrezali zahtevam. Agilne metodologije predvidevajo pregled med posameznimi iteracijami. Takšne preglede opravljajo razvijalci skupaj z naročnikom, ki je odgovoren za ocenjevanje uporabnosti in skladnosti z zahtevki. Tako Wright (2003) kot Erharuy (2007) razlagata, da XP to točko izpolnjujejo z uporabo programiranja v paru, saj takšen način razvoja zagotavlja, da en član ekipe neprestano nadzoruje drugega. Stålhane se z njima ne strinja v celoti in navaja, da programiranje v paru ni v skladu z definicijo sistematičnega pregleda kode, saj se osredotoča na kodiranje razvijalca, poleg tega pa programiranje v paru ne vključuje pregleda dokumentacije. Primera, ki ju obravnava Victoria (2004), sta zahteve te točke le delno izpolnjevala. Splošen načrt in izvedba je preverjala ekipa, ki je odgovorna za kakovost, podrobno načrtovanje pa ni bilo preverjano.

Podobno kot pri zahtevah o ustreznih rezultatih snovanja in razvoja je tudi pri skladnosti z zahtevami točke 7.3.4 veliko odvisno od izbire agilne metodologije in izvajanja le-te. V paru se lahko izvaja ves postopek razvoja, samo pisanje kode ali sploh nič. V slednjih dveh primerih mora podjetje zagotoviti postopke, da je rezultat vsake aktivnosti tudi pregledan.

Pogoj za skladnost z zahtevami točke 7.3.5 (overjanje načrtovanja in razvoja) je skladnost z zahtevami točke 7.3.1, v kateri Stålhane in Hanssen (2008) svetujeta splošno načrtovanje v okviru prve iteracije in podrobno načrtovanje pred vsako posamezno iteracijo. Erharuy (2007) navaja, da je podjetje z doslednim upoštevanjem XP metodologije skladno s to točko. Tako naročnik (oziroma njegov predstavnik) sproti preverja teste novih funkcionalnosti, ki dokazujejo, da so naročnikove zahteve izpolnjene s posameznimi funkcionalnostmi. Sprejemni testi so avtomatizirani, njihovi rezultati pa dokumentirani. Nawrocki et al. (2002) navaja več dobrih praks, ki zagotavljajo skladnost razvoja z XP metodologijo z ISO standardi. Te prakse so: pisanje testov pred pisanjem kode (angl. t*est – first coding*), celotna koda mora vsebovati teste posameznih modulov (angl. *unit test*), neprestana integracija (angl. *continuous integration*), optimizacija kode se opravi na koncu in za vsako najdeno napako se implementira test, ki zagotovi, da se napaka ne ponovi. Kljub vsem dobrim praksam sta bila primera, ki ju proučuje Victoria, le delno skladna s to točko. Ni bila zagotovljena sledljivost med uporabniško zgodbo, kodo, ki jo je implementirala, in testi, ki so delovanje te kode in uporabniške zgodbe testirali. Kot je razloženo pri točki 7.2, zagotovitev sledljivosti ne ovira agilnega razvoja, čeprav je le-ta ne predpisuje.

Naročnik opravi validacijo načrtovanja in razvoja (ki je zahtevana v točki 7.3.6,) po vsaki iteraciji, ko prejme novo verzijo izdelka. Zahteve te točke izpolnjuje vsako podjetje, ki se pri svojih projektih dosledno ravna po določeni agilni metodologiji.

Menim, da podjetje izpolni zahteve v zvezi z pregledom sprememb snovanja in razvoja, ki so navedene v točki 7.3.7, ko reši preostale zahteve točke 7.3. Erharuy (2007) navaja več dobrih praks, ki jih vsebujejo agilne metodologije dela. Te so: kratki intervali dobave programske opreme, redno preoblikovanje kode (angl. *refactoring*), funkcionalni avtomatizirani testi. Upoštevanje teh dobrih praks sicer še ne pomeni skladnosti s točko 7.3.7, se ji pa zelo približa.

4.2.4.4 (7.4) Nabava

Postopek nabave zadeva vse proizvode in storitve zunanjih izvajalcev, ki lahko vplivajo na izvajanje sistema vodenja kakovosti podjetja. Ta postopek vključuje tudi t.i. brezplačno programsko opremo, s katero si podjetje pomaga pri razvoju, kot del programskega paketa katerega izvedbo so prepustili podizvajalcem. Podjetje mora imeti določen postopek za nabavo izdelkov in specificiran način, kako določiti najprimernejšega dobavitelja (zahteve točke 7.4.1). Pred nabavo mora zbrati ustrezne informacije o izdelku. Če odda dela podizvajalcu, je priporočljivo pregledati sistem vodenja kakovosti podizvajalca (zahteve točke 7.4.2). Po nabavi produkta mora podjetje preveriti, ali je res dobilo zahtevano (zahteve točka 7.4.3).

Za izpolnjevanje zahtev te točke standarda ni važno, katero metodologijo razvoja programskega paketa uporablja podjetje. Uporaba agilnih metodologij pri razvoju ne vpliva

na izpolnjevanje zahtev, kakor tudi skladnost s to točko ne otežuje učinkovite uporabe katere od agilnih metodologij.

4.2.4.5 (7.5) Proizvodnja in izvedba storitev

Kot zahteva točka 7.5.1, je treba postopek proizvodnje in podpore planirati in izvršiti v nadzorovanih razmerah. Ta postopek obsega razvoj, dobavo, instalacijo, vzdrževanje programske opreme in tehnično podporo. Treba je ugotoviti ali med postopkom proizvodnje in podpore obstajajo deli, ki jih ni mogoče preveriti pred dobavo naročniku. Če obstajajo, je treba dokazati, da so ti postopki primerni in prinesejo pričakovani rezultat. Vzpostaviti je treba poseben nadzor nad takšnimi postopki (točka 7.5.2). Standardi v točki 7.5.3 zahtevajo enolično identifikacijo proizvoda. Spremembe med različicami izdelka je treba zabeležiti in posredovati naročniku (zahteva po sledljivosti vsake različice programskega paketa). Zahtevo po enolični identifikaciji proizvoda in njegovi sledljivosti uresničujejo s sistemom vodenja konfiguracij (angl. *Configuration management*). Kadar podjetje pri svojih procesih uporablja ali kako drugače nadzoruje lastnino naročnika, mora tej posvetiti posebno skrb. Kadar jo vključi v svoj izdelek, mora opraviti enake postopke, kot če bi jo kupovala (točka 7.5.4). Posebnih zahtev točke 7.5.5 o hranjenju proizvodov in sestavnih delov v procesu razvoja programske opreme ni težko izpolniti. Večinoma se nanašajo na fizično zaščito trdih diskov, zaščito pred virusi in na redno zagotavljanje varnostnih kopij.

Uporaba agilnih metodologij ne vpliva na skladnost z zahtevami te točke, niti skladnost s to točko ne otežuje učinkovite uporabe katere od agilnih metodologij.

4.2.4.6 (7.6) Obvladovanje nadzornih in merilnih naprav

Čeprav pri procesu razvoja programske opreme nadzorne in merilne opreme navadno ne uporabljajo neposredno, se v tem primeru ta standard nanaša na računalniško (strojno) opremo, ter na orodja, s katerimi preverjajo in zagotavljajo kakovost programske opreme (orodja za testiranje).

Uporaba agilnih metodologij ne vpliva na skladnost s to točko, niti skladnost s to točko ne otežuje učinkovite uporabe katere od agilnih metodologij. Ne glede na metodologijo dela, je treba pravilno poskrbeti za orodja za testiranje.

4.2.5 (8) Merjenje, analize in izboljševanje

Nadzorovanje in merjenje proizvodov, procesov, zadovoljstva odjemalcev in sistema vodenja ter zagotavljanje stalnega izboljševanja sistema so bistveni za vodenje sistema. Ta točka standarda je najpomembnejša, da podjetje napreduje in izboljšuje kakovost izdelkov ter povečuje učinkovitost procesa.

4.2.5.1 (8.1) Splošno

Ta točka je splošne narave in ne zahteva nobenega novega postopka. Ko je podjetje skladno z ostalimi točkami poglavij 4, 7 in 8, je skladno tudi z zahtevami te točke. Treba je zagotoviti nadzor nad procesi in analize le teh ter meritve zahtevanih lastnosti produkta, da lahko podjetje predstavi svoje izdelke kot skladne z zahtevami naročnika. Podjetje mora zagotoviti tudi neprestano izboljševanje nadzora, analiz procesa in meritev skladnosti produkta.

4.2.5.2 (8.2) Nadzorovanje in merjenje

Točka 8.2.1 zahteva, da se kakovost programske opreme meri s spremljanjem in merjenjem zadovoljstva odjemalcev in s spremljanjem njihovega občutka, da je podjetje zadovoljilo njihova pričakovanja. Standard v točki 8.2.2. zahteva, da mora podjetje zagotoviti notranjo revizijo v rednih časovnih intervalih. Namen te revizije je preverjanje, da sistem vodenja kakovosti izpolnjuje tako ISO standard kakovosti kot druge zahteve, ki si jih je podjetje postavilo, in da je sistem vodenja kakovosti izvajan in vzdrževan. Podjetje mora vzpostaviti primerne načine, metode za spremljanje in merjenje sistema vodenja kakovosti, kakor zahteva točka 8.2.3. S temi metodami pokaže podjetje svojo zmožnost, da s svojim procesom doseže planirane rezultate. Običajno se razmerje med planiranim in realiziranim časom, stroški za izvedbo določene aktivnosti in ravnijo kakovosti meri z določenimi kazalniki kakovosti. Prav tako mora podjetje, da zadovolji zahtevam točke 8.2.4, vzpostaviti ustrezne načine za spremljanje in merjenje lastnosti izdelka, da le ta izpolni zahteve. Običajno se meri lastnosti izdelka, kot so funkcionalnost, vzdržljivost, učinkovitost, prenosnost, uporabnost in zanesljivost.

Agilne metodologije ugotavljajo zadovoljstvo odjemalcev z izdelkom, ki je v delu, z neposredno komunikacijo z naročnikom po vsaki iteraciji in s komunikacijo z uporabniki zgodnjih različic aplikacije. Ker ta komunikacija ni dokumentirana in ni primerna za analizo, takšen način ugotavljanja zadovoljstva ne zadošča za skladnost z zahtevami točke 8.2.1. V podjetju morajo vzpostaviti metode za merjenje zadovoljstva odjemalcev, ki ga je mogoče kasneje analizirati. Menim, da je mogoče velik del zahtev v zvezi z merjenjem zadovoljstva odjemalcev izpolniti z uporabo orodij za podporo sledenju uporabniškim zgodbam, ki jih omenjam tudi pri točki 7.2.2. S takšnim orodjem bi lahko merili primera, ki ju podaja ISO 90003, 8.2.1 a) analiza podpore uporabnikom (na primer, koliko časa traja povprečen primer podpore, vrsta podpore…) in 8.2.1 d) potrebno število verzij po prvi verziji programskega paketa, da so odstranjene sporočene napake.

Izbira metodologije razvoja programske opreme ni bistvena za skladnost z zahtevo po notranjih presojah (točka 8.2.2). Uporaba agilnih metodologij ne vpliva na skladnost s to točko, niti skladnost s to točko ne otežuje razvoja s katero od agilnih metodologij.

Zahteve nadzorovanja in merjenja procesov (točka 8.2.3) izpolnjuje podjetje z rednimi pregledi dela po vsakem teku (angl. *Sprint review*) in z ocenjevanji procesa dela v daljših, vnaprej določenih intervalih (angl. *Retrospective*). Poleg teh, ki imajo podobno vlogo kot ocenjevanje procesa pri tradicionalnih metodologijah razvoja, podjetje sklene tudi, katere akcije bo uvedlo za bolj kakovostno in učinkovitejše delo. Tako izpolni tudi zahteve te točke, ugotavljata Stålhane in Hanssen (2008).

Za lažje ocenjevanje procesa dela obstaja več merljivih kazalnikov. Pri uporabi metodologije XP je smiselno meriti nadurno delo (XP zahtevajo, da razvijalci ne opravljajo nadur, vsaj ne dva tedna zapored), razpoložljivost predstavnika naročnika (njegova razpoložljivost je pomembna za učinkovito delo razvijalcev), hitrost izvajanja projekta (angl. *project velocity*), pogostost integracij (pogostost shranjevanja nove kode v projekt), način kodiranja za vsak zapis kode v projektu (programiranje v paru ali posamezno), hitrost programiranja (število vrstic, testnih scenarijev na uro…) in druge kazalnike (Nawrocki et al., 2002).

Pri pregledu praktičnih primerov, ki ju navaja Victoria (2004), ugotovimo, da sta opazovana primera le delno izpolnjevala zahteve standarda v točki 8.2.3, saj merjenje hitrosti izvedbe uporabniških zgodb (angl. *project velocity*) z uporabo točkovanja uporabniških zgodb ni dovolj za skladnost s to točko.

Menim, da mora podjetje za skladnost z zahtevami točke 8.2.3 izbrati primerne kazalnike, kakršne navajajo Nawrocki et al. (2002), jih spremljati in se na njihovi podlagi na pregledih tekov in ocenjevanjih procesa (angl. *Retrospective*) odločati o ukrepih za izboljšanje delovnega procesa.

Zahteve o nadzorovanju in merjenju proizvodov (točka 8.2.4) podjetje izpolni z rednimi pregledi po vsaki končani iteraciji. Delovanje vsake uporabniške zgodbe je predstavljeno v prisotnosti naročnika in lahko tudi vodstva. Ustreznost posameznih funkcionalnosti je zabeležena v spisku vseh uporabniških zgodb (angl. *product backlog*) ali podobnem dokumentu (Stålhane & Hanssen, 2008; Erharuyi, 2007). Na teh sestankih se predstavi poročila opravljenih testov enot (angl. *unit tests*) in sprejemnih testov (angl. *acceptance tests*). Obe vrsti testov sta avtomatizirani in preverjata tako funkcionalnosti kot performance (Nawrocki et al., 2002). Praktična primera, ki ju raziskuje Victoria sta bila le delno skladna s to točko standarda, saj pri projektih vse do zadnje iteracije niso preverjali in beležili rezultatov nefunkcionalnih zahtev. Kot že pri več točkah tudi tu opazimo razkorak med teorijo in prakso, saj so v praksi nefunkcionalne zahteve za posamezno funkcionalnost pogosto testirane šele proti koncu projekta. Podjetje mora zagotoviti, da se to ne zgodi in da ima naročnik možnost potrditi ali ovreči vsak vidik določene funkcionalnosti, ko je končana.

4.2.5.3 (8.3) Obvladovanje neskladnih proizvodov

Podjetje mora zagotoviti, da izdelek ali njegov del, ki ni v skladu s standardi, identificirajo in primerno spremljajo, da ne pride do nehotene uporabe takšnega izdelka. Vzpostavljeni morajo biti ustrezni postopki glede neskladnih produktov, ki so primerno dokumentirani in vzdrževani.

Prav uporaba XP naj bi zagotovila, da naročnik ne prejme neskladnih proizvodov, zato tudi ne narekuje postopka za nadaljnjo upravljanje s takšnimi proizvodi. Tako tudi v primerih, ki ju opisuje Victoria (2004) niso imeli postopkov za upravljanje z neskladnimi proizvodi. Neskladne dele kode se namreč zavrže. Postopek, da do neskladnih funkcionalnosti ne pride, natančneje opisuje Erharuyi (2007). Razvijalci v sodelovanju z naročnikom sestavijo teste, ki zagotavljajo, da bo funkcionalnost izpolnila naročnikove zahteve. Pred integracijo kode za novo funkcionalnost v projekt morajo biti ti testi opravljeni.

Ne glede na to, da podjetje z uporabo katere od agilnih metodologij razvoja vpelje tudi postopke, ki preprečujejo, da bi naročnik prejel neskladne proizvode, mora za skladnost z zahtevami točke 8.3 vpeljati tudi postopke za primer, ko naročnik kljub vsemu prejme neskladen proizvod. To so postopki, da se minimizira morebitna škoda, ki bi jo neskladni proizvodi lahko povzročili, in korektivni postopki, da se dobava neskladnih proizvodov ne ponovi v prihodnje. V podjetju morajo tudi zagotoviti, da morebitna dopuščena odstopanja potrdi pooblaščena oseba pri odjemalcu.

4.2.5.4 (8.4) Analiza podatkov

Podjetje mora zbrati in analizirati tiste podatke, s katerimi lahko dokaže ustreznost in učinkovitost sistema vodenja kakovosti in oceni področja, na katerih lahko sistem vodenja kakovosti izpopolni.

Izbira metodologije razvoja programske opreme ne vpliva na skladnost s to točko standarda, saj mora podjetje uporabiti enake postopke, kot bi jih pri tradicionalnih metodologijah. Agilne metodologije ne predvidevajo posebnih postopkov za analizo naročnikovega zadovoljstva, saj se razvijalci o tem seznanijo z neposredno komunikacijo.

4.2.5.5 (8.5) Izboljševanje

Podjetje mora neprestano izpopolnjevati sistem vodenja kakovosti. Strateški vidik te zahteve je doseči proces, ki nas sili v izboljševanje (točka 8.5.1). Kadar se pojavijo produkti ali postopki, ki niso v skladu s standardom, je treba vpeljati ustrezne korektivne postopke in ukrepe, ki poskrbijo, da do neustreznosti v prihodnje ne bo prišlo (točka 8.5.2). Podjetje mora raziskati kritična področja, kjer se lahko pokaže nevarnost, da izdelki ali postopki podjetja ne

bi bili več skladni s standardom in uvesti ustrezne preventivne postopke, da prepreči potencialne probleme.

Agilne metodologije temeljijo na nenehnem izboljševanju tako postopkov dela kot znanja in sposobnosti razvijalcev. Ustrezna dobra praksa pri uporabi metodologije Scrum, s katero se v podjetju nenehno izboljšujejo, je uvedba spiska predlogov za izboljšave, ki ga lahko vsak član ekipe dopolnjuje. Na začetku vsake iteracije člani glasujejo, katero izboljšavo bodo uvedli v naslednji iteraciji. Victorieva (2004) primera kljub rednim sestankom in tudi uvedbi izboljšav v postopkih dela te točke nista v celoti izpolnila, saj rezultatov izboljšav nista dokumentirala. Ugotovimo lahko, da je treba za skladnost z točko 8.5.1 poleg izvajanja omenjene dobre prakse spremljati tudi rezultate vpeljanih izboljšav in jih dokumentirati.

Podjetje mora sprejeti ustrezne ukrepe, da se odstranijo razlogi, ki so povzročili neskladnost produkta (točka 8.5.2 – korektivni ukrepi), ali bi takšno neskladnost lahko povzročili (točka 8.5.3 – preventivni ukrepi). Neskladnosti produkta ugotovijo ob pregledu iteracije. Tako Nawrocki et al. (2002) kot Erharuyi (2007) navajajo, da je korektivni ukrep, ki uresničuje zahteve te točke standarda, posodabljanje testov. S tem se ne strinjata Stålhane in Hanssen (2008). Trdita, da je treba ugotoviti in odpraviti vzrok, da določen test ni preveril vseh zahtev, ko je bil sestavljen, in ga odpraviti. Za neskladnost produkta navajata štiri možne razloge: premalo sredstev (na primer informacij pred začetkom iteracije), nepredvidene težave, nezadostno razumevanje zahtevkov in slab proces. Le zadnji omenjeni razlog je smiseln za obravnavo v teh dveh točkah. Ugotovljene slabosti odpravimo enako, kot velja za skladnost z zahtevo točke 8.5.1.

4.3 POVZETEK

V tem poglavju želim poudariti najpomembnejše ugotovitve četrtega poglavja, ki govori o uvajanju standarda ISO 9001 v podjetje, ki pri razvoju programske opreme uporablja agilne metodologije. Čeprav ISO 90003 posebej navaja, da ISO 9001 ne sugerira uporabe določene metodologije razvoja programske opreme, je iz vsebine nekaterih zahtev standarda razvidno, da ob pripravi standarda ISO 9001:2000 agilne metodologije še niso bile širše uporabljane in jih avtorji standarda niso predvideli kot življenjski ciklus razvoja. Kasnejši ISO 9001:2008 pa posameznih zahtev ni bistveno spremenil.

Opazil sem tudi, da morajo podjetja, ki delajo izdelek za neznanega naročnika, svoj delovni proces manj prilagajati in zato laže vpeljejo standard ISO 9001, kot tista, ki delajo za znanega naročnika oziroma se prijavijo na razpis za izvedbo določenega projekta. Pri razvoju za neznanega naročnika vsa komunikacija o bodočem izdelku poteka znotraj podjetja, kar olajšuje potrjevanje pravilnosti narejenega.

Ob branju poglavij 4.1 in 4.2 je treba upoštevati, da jemljem za izhodišče podjetje, ki je po svoje idealno in pri svojih projektih izvaja vse dobre prakse agilnih metodologij Scrum in

XP. Ker je le redkokatero podjetje sposobno določeno metodologijo vpeljati v celoti, tako kot je mišljena, primeri iz prakse pogosto niso skladni z določenimi zahtevami standarda, čeprav bi jih lahko izpolnjevali z doslednim izvajanjem dobrih praks izbrane metodologije. V takšnih primerih je lahko vpeljava standarda kakovosti ISO 9001 zelo koristna. Če se v podjetju odločijo, da bodo določeno zahtevo standarda izpolnjevali z uporabo določene dobre prakse agilne metodologije, s tem zagotovijo ali vsaj povečajo možnosti, da se bodo te prakse tudi držali.

Veliko dobrih praks agilnih metodologij XP in Scrum je zelo dobra podlaga (potrebne so namreč razmeroma majhne spremembe oziroma nadgradnje načina dela), da je podjetje skladno z več zahtevami standarda ISO 9001. Navedel bom najbolj splošne nadgradnje načina dela, ki jih je priporočljivo uporabiti ne glede na specifiko posameznega podjetja in njegovih projektov.

Pomembno je, da v podjetju uporabljajo določeno orodje za podporo agilnim metodologijam. Na tak način v podjetju prenesejo tablo, katere primer je na Sliki 6 in na kateri je zapisan plan določenega teka, v digitalizirano obliko. Takšno orodje precej olajša sledenje posameznim naročnikovim zahtevam, da se za vsako zahtevo ve, kdo in kdaj jo je naročil, definiral, posodobil zahteve, implementiral, preveril, verificiral ter v kakšnem statusu je trenutno. S takšnim orodjem je mogoče tudi meriti nekatere kazalce procesa, ki vplivajo na zadovoljstvo odjemalcev, na primer hitrost, s katero se podjetje odzove uporabniku in izpolni njegovo povpraševanje.

Dobro orodje za podporo sledenju, ki jih je na trgu zelo veliko, tudi močno olajšuje označevanje različic izdelka (angl. *configuration controll*). Seveda je od posameznega podjetja odvisno, kako izkoristi orodje, ki ga uporablja, vendar si brez uporabe takšnega orodja ne predstavljam, da bi lahko podjetje vpeljalo standard ISO 9001 in hkrati ohranilo agilnost pri razvoju.

Pri popisovanju zahtev morajo v podjetju zagotoviti, da med lastnosti posamezne naročnikove zahteve zabeležijo tudi nefunkcionalne zahteve. Za vsako implementirano zahtevo mora biti obvezno zabeleženo, da je naročnik potrdil, da je dobil tisto, o čemer so se dogovorili. Naročnik mora imeti možnost potrditi ali ovreči vsak vidik določene funkcionalnosti, ko je ta končana, tudi nefunkcionalne vidike. Pred izvedbo določenega sklopa mora biti razvijalcem jasno, kakšen bo rezultat, in to mora biti mogoče dokazati. Za te zahteve metodologija XP sicer predvideva določene dobre prakse, z uporabo katerih je podjetje lahko skladno s standardom ISO 9001, vendar jih podjetja ponavadi izpolnijo le delno, zato jih tu še posebej poudarjam.

Sestanke, na katerih sprejemajo odločitve ali potrjujejo pravilnost implementiranega (pri Scrumu so to sprint review, sprint planning in retrospective), morajo v podjetju redno snemati ali o njih napisati zapisnike, da so odločitve in potrditve lahko ustrezno arhivirane in po

potrebi dosegljive. Na teh sestankih se pogosto sprejemajo odločitve za izboljšanje sistema vodenja kakovosti, ki ga ima podjetje. Metode za sprejemanje takšnih odločitev imata tako Scrum, kot XP. Ker metodologiji ne predpisujeta postopkov, da se rezultate teh izboljšav tudi spremlja, mora podjetje takšne postopke uvesti samo.

Poleg navedenih nadgradenj dobrih praks XP in Scruma, ki učinkovito izpolnjujejo več zahtev standarda ISO 9001, obstajajo zahteve za izpolnitev pri katerih izbira metodologije razvoja programske opreme ni bistvena. Primerov, kako podjetje izpolni takšne zahteve ne navajam, saj to ni tema te naloge. Opozoril pa bi na dve zahtevi, o katerih menim, da ju je v nekaterih primerih teže izpolniti in hkrati ohraniti agilnost, kakršno omogoča uporaba agilnih metodologij razvoja.

Točka 5.5.2 zahteva, da je le ena oseba odgovorna za kakovost in poznavanje naročnikovih zahtev. Ker so pri agilnih metodologijah za kakovost odgovorni vsi člani razvojne ekipe, mora podjetje najti ustrezen način, da več kot le formalno izpolni zahtevo ter hkrati ohrani občutek odgovornosti za kakovostne rezultate dela vseh članov.

Točka 7.2.2 zahteva, da podjetje podrobno pregleda naročnikove zahtevke, preden se zaveže za dobavo produkta. Strogo upoštevanje te zahteve onemogoča uporabo agilnih metodologij razvoja programske opreme ob hkratni skladnosti s standardom ISO 9001. Dobro je, da presojevalec razume razvoj z uporabo agilnih metodologij. Podjetje namreč lahko dokaže, da je določen končni rezultat sposobno doseči, podrobno definiranje vsake zahteve pred začetkom razvoja pa bi izničilo bistvo agilnega razvoja.

Menim, da je lahko zahteva glede pregleda naročnikovih zahtev najbolj problematična za izpolnitev v podjetju, ki razvija programsko opremo za znanega naročnika z uporabo določene agilne metodologije, seveda odvisno od specifik posameznega podjetja. Omenil bi še tri zahteve, na katere morajo biti posebej pozorni v podjetju, ki je že skladno s standardom ISO 9001, pa želijo začeti uporabljati določeno agilno metodologijo.

Važno je, da se vodstvo zaveda in podpira sistem vodenja kakovosti, ki ga ima podjetje, o čemer govori tudi točka 5.3. Idejo o uporabi določene agilne metodologije sicer lahko da razvojni oddelek, vendar bo uporaba metodologije težko zaživela, če je vodstvo ne podpira. Uporaba agilnih metodologij pri razvoju namreč pogosto ne zadeva samo razvoja izdelka.

Da pogodba ustreza načinu razvoja, mora biti v njej zajeto, kako bo naročnik sodeloval pri razvoju, zato je v podjetju pomembno sodelovanje prodajnega oddelka. Naročnik, s katerim se ta oddelek pogovarja na začetku, mora biti namreč pripravljen na večje sodelovanje tekom razvojem. Pomembna je tudi dobro definirana vloga predstavnika naročnika, o kateri govori točka 7.2.2.3 v ISO 90003.

Pomembno je tudi sodelovanje kadrovskega in drugih oddelkov. Potrebe, katerih izpolnjevanje zahtevata točki 6.2 in 6.3 v standardu ISO 9001, se pri agilnem razvoju namreč razlikujejo od tradicionalnih potreb. Posamezni razvijalec mora biti nagrajen tako, da je stimuliran k timskemu delu in ne tekmovalno. Ravno tako se lahko pojavijo potrebe po spremenjenih prostorskih rešitvah in drugačni opremi.

SKLEP

Razvoj programske opreme je v večini primerov neučinkovit, saj uporabniki uporabljajo je le manjši del razvitih funkcionalnosti. Le redke izdelke programske opreme lahko označimo kot kakovostne, uporabniki novih aplikacij pa so pogosto razočarani, saj nove različice ne izpolnijo vseh njihovih pričakovanj.

Kakovost programske opreme se v zadnjih desetih letih izboljšuje in zagovorniki agilnih metodologij razvoja programske opreme menijo, da je vzrok za napredek vedno pogostejša uporaba uporablja agilne metodologije razvoja programske opreme. Cilj agilnih metodologij, torej redna dobava uporabne programske opreme, je dosežen predvsem z večjim sodelovanjem naročnika in odstranjevanjem nepotrebnega dela iz procesa razvoja. Ravno odstranjevanje nepotrebnega dela, ki v veliki meri nanaša na zmanjšanje dokumentacije, pa je marsikatero podjetje pripeljalo do novega izziva. Hkratna uporaba katere od agilnih metodologij in skladnost z ISO standardom kakovosti, ki prinaša številne druge prednosti za podjetje, je namreč na videz nemogoča.

Tema pričujoče diplomske naloge je podrobna primerjava zahtev standarda ISO 9001 in lastnosti razvoja z agilnimi metodologijami. Pokaže, da je večina pomislekov za hkratno uporabo agilnih metodologij in skladnosti s standardom kakovosti posledica nerazumevanja ene od obeh tematik, nekatere zahteve standarda pa je v resnici teže izpolniti kot z uporabo tradicionalnih metodologij razvoja programske opreme.

Podjetje, ki uporablja katero od agilnih metodologij razvoja programske opreme, mora posebej paziti, da izdeluje zahtevano dokumentacijo kot naravni element procesa razvoja in ne izključno zaradi skladnosti s standardom. Glede na skladnost s standardi mora podjetje dokazati, da je metodologija razvoja, ki jo uporablja, primerna da z njo doseže želene rezultate in da to metodologijo dosledno uporablja. Kljub premišljeni implementaciji standarda ISO 9001 pa si olajša delo, če je pozorno pri izbiri presojevalca: ta naj razume tudi agilni razvoj programske opreme.

Pričujoče diplomsko delo predstavlja smisel in nekatere značilnosti razvoja programske opreme z agilnimi metodologijami in osnovne značilnosti standarda ISO 9001. Z opisi raziskav sorodne tematike bralcu prikaže sliko, kaj o hkratni uporabi ISO standarda kakovosti in agilnih metodologij razvoja menijo drugi avtorji in ga napoti na literaturo, ki je le najbližje primeru, ki ga zanima. Jedro diplomskega dela je pregled zahtev standarda po posameznih

točkah. Za vsako točko, kjer je to potrebno, navaja, kako jo je mogoče izpolniti ob hkratni uporabi agilnih metodologij in kje lahko nastopijo težave.

Menim, da sem dosegel cilj, ki sem si ga postavil za diplomsko nalogo in da je zanimiva tako za podjetja, ki že uporabljajo določeno agilno metodologijo in želijo biti skladna s standardom ISO 9001, kot za podjetja, ki so že skladna s standardom in imajo namen vpeljati bolj agilni proces razvoja programske opreme.

VIRI IN LITERATURA

1. Barens, F. (2000). Good Business Sense Is the Key to Confronting ISO 9000. *AllBusiness.com, Inc*. Najdeno 16. januarja 2011 na spletnem naslovu http://www.-allbusiness.com/specialty-businesses/713376-1.html
2. Beck, K. (1999). *Extreme Programming Explained: Embarace Change*. Boston: Addison-Wesley.
3. Boehm, B., & Turner, R. (2004). Balancing Agility and Discipline: Evaluating and Integrating Agile and Plan-Driven Methods. Zbornik *26th International Conference on Software Engineering (ICSE'04)*. Washington DC: IEEE Computer Society.
4. Cohn, M. (2010). *Succeeding with Agile – Software Developement Using Scrum*. Boston: Addison-Wesley.
5. Crispin, L. & Gregory J. (2009). *Agile Testing – A Practical Guide for Testers and Agile Teams*. Boston: Addison-Wesley.
6. Crystal Clear (software development) (b.l.). V *Wikipedia the free encyclopedia.* Najdeno 15. decembra 2010 na spletnem naslovu: http://en.wikipedia.org/wiki/-Crystal_Clear_ (software_development)
7. Dynamic Systems Development Method. (b.l.). V *Wikipedia the free encyclopedia*. Najdeno 15. decembra 2010 na spletnem naslovu http://en.wikipedia.org/wiki/-Dynamic_Systems_Development_Method
8. Erharuyi, E. (2007). *Combining eXtreme Programming with ISO 9000:2000 to Improve Nigerian Software Development Processe*. Ronneby: School of Engineering Blekinge Institute of Technology.
9. Extreme_Programming. (b.l.). V *Wikipedia the free encyclopedia*. Najdeno 10. oktobra 2010 na spletnem naslovu http://en.wikipedia.org/wiki/Extreme_-Programming
10. Feature Driven Development. (b.l.). V *Wikipedia the free encyclopedia*. Najdeno 15. decembra 2010 na spletnem naslovu http://en.wikipedia.org/wiki/Feature_Driven_ Development
11. Gilb, T. (2005). *Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage*. Oxford: Elsevier Butterworth-Heinemann.
12. Gregory, J. (2010). *Quality – What is it, and what's the cost?* Najdeno 3. oktobra 2010 na spletnem naslovu http://janetgregory.blogspot.com/
13. Heras, I., Dick, G., & Casadesús, M. (2002). *ISO 9000 registration's impact on sales and profitability*. Bingley: Emerald Group Publishing Limited.
14. *History: The Agile Manifesto*. (2001). Najdeno 7. januarja 2011 na spletnem naslovu http://agilemanifesto.org/history.html
15. Hoyle, D. (2009). *ISO 9000 Quality Systems Handbook - updated for the ISO 9001:2008 standard* (6th Edition). Oxford: Elsevier Ltd.

16. Internal Organization for Standardization. (2004). *ISO/IEC 90003:2004 (Software engineering – Guidelines for the application of ISO 9001:2000 to computer software)*. Geneve: International Organization for Standardisation.
17. Internal Organization for Standardization. (2009a). *ISO 9000 – Selection and use*. Geneve: International Organization for Standardisation.
18. Internal Organization for Standardization. (2009b). *The ISO Survey – 2009*, najdeno 16. januarja 2011 na spletnem naslovu http://www.iso.org/iso/survey2009.pdf
19. Internal Organization for Standardization. (2011a). Q*uality management principles*. Najdeno 5. februarja 2011 na spletnem naslovu http://www.iso.org/iso/iso_catalogue/-management_standards/quality_management/qmp.htm
20. Internal Organization for Standardization. (2011b). *Understand the basics*. Najdeno 3. novembra na spletnem naslovu http://www.iso.org/iso/iso_catalogue/management_-standards/management_system_basics
21. Johnson, J. (2002). *Build only the Features You Need*. Najdeno 4. oktobra 2010 na spletnem naslovu http://martinfowler.com/articles/xp2002.html
22. Kniberg, H. (2007). *Scrum and XP from Tranches*. C4Media Inc. najdeno 12. aprila 2011 na spletnem naslovu http://www.crisp.se/henrik.kniberg/ScrumAndXpFromThe-Trenches.pdf
23. Kniberg, H. (2010, 8. oktober). *The Essence of Agile* [predavanje na konferenci]. Najdeno 4. aprila 2011 na spletni strani http://agileee.org/2010/ Kyiv: Agile East Europe Conference.
24. *Manifesto for Agile Software Development*. Najdeno 2. avgusta 2010 na spletnem naslovu http://www.agilemanifesto.org/
25. McMichael, B., & Lombardi, M. (2007). ISO 9001 and Agile Development. *Zbornik Agile 2007 Conference.* Washington: IEEE Computer Society Washington
26. McBreen, P. (2003). *Questioning Extreme Programming.* Boston: Pearson Education
28. *Mountain Goat Software.* Najdeno 10. oktobra na spletnem naslovu http://www. mountaingoatsoftware.com/scrum/figures
29. Namioka, A., & Bran, C. (2004). eXtreme ISO. *Zbornik konference OOPL'04*, Vancouver: ACM Sigplan
30. Nanda, V. (2003). *Achieving Compliance and Continous Improvement in Software Development Companies*. Milwauke: ASQ Quality Press
31. Nawrocki, J. R., Jasiński, M., Walter, B., & Wojciechowsk A. (2002). Combining Extreme Programming with ISO 9000. *Zbornik EurAsia-ICT 2002: Information and Communication Technology First EurAsian Conference Shiraz*, Iran: October 29–31. Springerlink.
32. Pivka M., & Košir A. (2010). *ISO 9001 pri razvoju in vzdrževanju programske opreme ter uporaba smernic ISO/IEC 90003:2004*. Ljubljana: SIQ
33. Poppendieck, M. (2007). *The Challenges of Bringing Lean to Software Development*. Najdeno 15. decembra 2010 na spletni strani http://www.leanessays.com/2007/05/-challenges-of-bringing-lean-to-software.html

34. Poppendick, M. (2010). *It's Not About Software*. [predavanje na konferenci]. Najdeno 4. aprila 2011 na spletni strani http://agileee.org/2010/ Kyiv: Agile East Europe Conference.
35. *Principles behind the Agile Manifesto.* (2001). Najdeno 7. januarja 2011 na spletni strani http://agilemanifesto.org/principles.html
36. Schwaber, K., & Beedle, M. (2002). *Agile Software Development with Scrum*. New Jersey: Prentice Hall
37. Scrum. (b.l.). V *Wikipedia the free encyclopedia*. Najdeno 10. oktobra 2010 na spletnem naslovu http://en.wikipedia.org/wiki/Scrum_(development)
38. Shore, J., & Warden, S. (2007) *The art of Agile Developement*. Sebastopol: O'Reilly.
39. Slovenski inštitut za standardizacijo. (2005). *Slovenski standard. SIST EN ISO 9000, Sistemi vodenja kakovosti. Osnove in slovar (ISO 9000:2005) : (istoveten EN ISO 9000:2005)*. Ljubljana: Slovenski inštitut za standardizacijo (SIST)
40. Slovenski inštitut za standardizacijo – SIST. (2011). *Sistem vodenja kakovosti in slovenski standard sist en ISO 9001:2008*, Najdeno 16. Januarja 2011 na spletni strani http://www.sist.si/index.php?option=com_content&view=article&id=112&catid=39Itemid=161
41. Stålhane, T., & Hanssen, G. K. (2008). The application of ISO 9001 to agile software development. *Zbornik 9. mednarodne konference PROFES 2008 Monte Porzio Catone*. Rome: Springer
42. Stephanie, C., (2005). *So many standards to follow, so little payoff*. Inc. Newsletter. Najdeno 16. januarja 2011 na spletnem naslovu http://www.inc.com/magazine/2005-0501/management.html
43. Stephens, M., & Rosenberg, D. (2003). *Extreme Programming Refactored: The Case Against XP*. New York: Apress
44. The Standish Group Report. (1995). *CHAOS Summary 1995*. Najdeno 4. oktobra 2010 na spletnem naslovu http://www.projectsmart.co.uk/docs/chaos-report.pdf
45. The Standish Group Report. (2009). *CHAOS Summary 2009*. Najdeno 4. oktobra 2010 na spletnem naslovu http://www1.standishgroup.com/newsroom/chaos_2009.php
46. V-Model (software development). (b. l.) V *Wikipedia the free encyclopedia* Najdeno 7. januarja 2011 na spletnem naslovu http://en.wikipedia.org/wiki/V-Model-_(software_development)
47. Victoria, D. (2004). *Aligning XP with ISO 9001:2000 –TickIT Guide 5.0 - A case study in two academic software projects*. School of Engineering Blekinge Institute of Technoloe
48. Vriens, C. (2003). Certifying for CMM Level 2 and ISO9001 with XP@Scrum. *Konferenca Agile Aliance 2003*. Washington: IEEE Computer Society Washington
49. Waterfall model. (b.l.). V *Wikipedia the free encyclopedia*. Najdeno 7. januarja 2011, na spletnem naslovu http://en.wikipedia.org/wiki/Waterfall_model
50. Wright, G. (2003). Achieving ISO 9001 Certification for an XP Company. *Zbornik Extreme Programming and Agile Methods – XP/AgileUniverse 2003*. Chichago: Springer