UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

NATAŠA ŽABKAR

MODEL ZA SPREMLJANJE UČINKOVITOSTI AGILNEGA RAZVOJA PROGRAMSKE OPREME

DOKTORSKA DISERTACIJA

MENTOR: IZR. PROF. DR. VILJAN MAHNIČ

Ljubljana, 2013

POVZETEK IN KLJUČNE BESEDE

Agilne metodologije so prisotne več kot deset let in v tem času so se uveljavile kot učinkovite, čeprav je število empiričnih raziskav na tem področju skromno, predvsem na področju spremljanja učinkovitosti agilnega razvoja programske opreme.

Najbolj razširjena agilna metodologija *Scrum* pri spremljanju učinkovitosti uporablja samo en kazalnik in sicer obseg preostalega dela za realizacijo *Uporabniške zgodbe* (ang. User Story) iz *Seznama Uporabniških zgodb* (ang. Product Backlog) ali pa za dokončanje *Naloge* iz *Seznama Nalog Iteracije* (ang. Sprint Backlog). Sčasoma se je pokazala potreba po dopolnitvi tega kazalnika. V literaturi navajajo, da deležniki pogrešajo metrike glede predvidenega zaključka projekta [Sch05] ter, da obstaja potreba po boljšem merjenju vrednosti glede na stroške [Yap06], pri čemer je potrebno upoštevati, da metrike, ki ne upoštevajo načel agilnosti, lahko povzročijo ravno nasproten učinek [Har06].

Namen doktorske disertacije je bil razviti model za spremljanje učinkovitosti agilnega razvoja programske opreme, ki bo skladen z načeli agilnosti.

Z metodo teoretičnega raziskovanja smo povzeli raziskovalno in strokovno literaturo s področja razvoja programske opreme, projektnega vodenja, agilnega razvoja programske opreme in metodologije *Scrum.* Najprej smo na področju razvoja programske opreme predstavili standarde SWEBOK, ISO/IEC 12207: »Informacijska tehnologije - Procesi življenjskega cikla programske opreme«, Združeni zmožnostno zrelostni model (CMMI) ter ogrodje COBIT za upravljanje informacijske tehnologije iz vidika razvoja programske opreme. Potem smo na področju projektnega vodenja predstavili standarde PMBOK ter ogrodje COBIT za upravljanje informacijske tehnologije iz vidika projektnega vodenja. Na področju agilnega razvoja programske opreme smo predstavili agilni razvoj programske opreme, primerjali agilne in slapovne metodologije, podali opis metodologije *Scrum* ter opis vlog pri metodologiji *Scrum*.

Teoretično raziskovanje smo nadaljevali na področju merjenja učinkovitosti razvoja programske opreme z uporabo agilnih metod, zrelosti agilnih metod, opredelitve metrik, poročanja o napredku in modelih za spremljanje učinkovitosti. Rezultati raziskovanja so potrdili, da do sedaj še ni bil razvit model za spremljanje učinkovitosti agilnega razvoja programske opreme, ki bi na sistematičen način omogočil spremljanje agilnih projektov, upoštevajoč tradicionalne metrike, kot so prislužena vrednost in agilne metrike, kot je hitrost ter omogočil enostavno in pregledno poročanje, upoštevajoč načela agilnosti glede zbiranja podatkov. Odločili smo se, da bomo pri razvoju našega modela uporabili koncept sistema PPS (Process Performance Measurement System) [Kue00], ki je procesno usmerjen in učinkovitost opredeljuje kot zadovoljstvo vseh deležnikov procesa ter neprekinjeno izboljševanje procesa.

Pri razvoju našega modela smo upoštevali dosežke Laboratorija za tehnologijo programske opreme na Fakulteti za računalništvo in informatiko Univerze v Ljubljani, ki na tem področju izvaja raziskave od vključno leta 2005, ko je bilo ogrodje *Scrum* prvič uporabljeno pri razvoju

aplikativne programske opreme »e-študent«. Delovne različice modela so bile v študijskih letih po letu 2008 uporabljene v akademskem okolju pri poučevanju predmeta »Tehnologija programske opreme«, v letu 2011 pa tudi v gospodarskem okolju. Izkušnje ob dosedanji uporabi modela smo vgradili v tekočo različico modela, ki smo ga poimenovali *AGIT* in je opisana v doktorski disertaciji. Nadaljnji razvoj modela *AGIT* bo potekal v okviru Laboratorija za tehnologijo programske opreme.

Ključni prispevek doktorske disertacije k znanosti je izdelava celovitega modela *AGIT* za spremljanje učinkovitosti agilnega razvoja programske opreme. V okviru modela za spremljanje učinkovitosti agilnega razvoja programske opreme so prispevki k znanosti:

- opredelitev ključnih kazalnikov za spremljanje učinkovitosti;
- opredelitev metrik, potrebnih za izračun ključnih kazalnikov;
- opredelitev postopkov za zbiranje podatkov;
- opredelitev organizacije podatkov in
- opredelitev oblike poročanja.

Model *AGIT* smo razvili po korakih, ki sledijo konceptu PPS. Najprej smo opisali razvojni proces razvoja programske opreme z uporabo ogrodja *Scrum*. Potem smo določili deležnike razvojnega procesa: Poslovodstvo, *Lastnik izdelka*, *Scrum Mojster* in *Razvojno moštvo*. Za vsakega izmed deležnikov smo določili cilje, za vsakega izmed ciljev pa kazalnike učinkovitosti. Nato smo podali podroben opis kazalnikov učinkovitosti za posameznega deležnika. Sledila je opredelitev ciljnih in mejnih vrednosti kazalnikov učinkovitosti. Potem smo določili način zbiranja podatkov, pri čemer smo sledili agilnim načelom ter oblikovali podatkovni model. Na koncu smo določili tudi način prikazovanja podatkov.

Uporabnost modela *AGIT* smo preverili na študijskem primeru. V doktorski disertaciji smo najprej predstavili podjetje in projekt ter podmnožico kazalnikov modela *AGIT*, ki smo jo uporabili pri preverjanju modela in predstavili z uporabo naslednjih diagramov: *Diagram spremembe Hitrosti, Padajoči diagram spremljanja napredka Izdaje, Padajoči diagram spremljanja napredka Iteracije, SPI diagram, CPI diagram* in *BRE tabela*. Primer smo opisali iz časovnega, organizacijskega in kakovostnega vidika. Predstavili smo potek *Izdaje* in *Iteracij* ter rezultate projekta. Sledila je predstavitev rezultatov merjenja učinkovitosti na ravni *Izdaje* ter na ravni posamezne *Iteracije*. Podali smo tudi rezultate analize točnosti ocen zahtevnosti *Uporabniških zgodb*. Ob preverbi smo ugotovili, da je model AGIT skladen z načeli agilnosti glede načina zbiranja in prikazovanja podatkov.

Upoštevajoč rezultate preverbe smo v zaključku doktorske disertacije podali smernice za nadaljnje raziskave.

Ključne besede:
Agilni razvoj programske opreme, AGIT, CMMI, COBIT, Kazalniki učinkovitosti, Merjenje učinkovitosti, Načrtovalni poker, Projektno poročanje, PMBOK, Scrum, Seznam Uporabniških zahtev, Seznam Nalog Iteracije, Spremljanje učinkovitosti, SWEBOK, Iteracija, Izdaja. Učinkovitost razvoja programske opreme, Uporabniška zgodba

ABSTRACT AND KEYWORDS

Agile methodologies have been in use for more than ten years and during this time they proved to be efficient, even though number of empirical research is scarce, especially regarding agile software development performance monitoring.

The most popular agile framework *Scrum* is using only one measure of performance: the amount of work remaining for implementation of *User Story* from the *Product Backlog* or for implementation of *Task* from the *Sprint Backlog*. In time the need for additional measures was acknowledged. According to [Sch05], the stakeholders lack the metrics regarding the finishing project date and better measurement of value in comparison with costs [Yap06]. [Har06] states that the agility principles should be taken into account when adding new measures or the effect can be the opposite [Har06].

The purpose of Doctoral Dissertation was to develop the model for agile software development performance monitoring in compliance with agility principles.

Theoretical research was performed to summarize research and practitioners literature in the areas of software development, project management, agile software development and *Scrum* framework. In the area of software development the following standards have been analyzed: SWEBOK, ISO/IEC 12207: "Information technology – Software life cycle processes", Capability Maturity Model Integration (CMMI) and COBIT framework from the software development aspect. In the area of project management the following standards have been analyzed: PMBOK and COBIT framework from the project management aspect. In the area of agile software development a definition of agile software development has been presented, followed by comparison between agile and waterfall methodologies, the *Scrum* framework presentation and the *Scrum* role description.

Further theoretical research has been performed regarding performance monitoring in the areas of agile methods and their maturity, metrics definition, progress reporting and performance monitoring models. The results of this research confirmed the lack of agile software development performance monitoring model that would systematically enable agile project monitoring while using traditional metrics, such as earned value and agile metrics, such as *Velocity,* while providing simple and transparent reporting and complying with agile principles for collecting and presenting data. Due to its process orientation the concept of PPMS (Process Performance Measurement System) [Kue00] has been selected for the framework for the model development process. The PPMS defines process performance as a degree of stakeholder satisfaction of all stakeholders and process improvement.

The resulting *AGIT* model includes the research results from the Software Technology Laboratory at the Faculty of Computer and Information Science at the University of Ljubljana since year 2005, when they first used *Scrum* framework for the application software development called "e-student". Working versions of the model have been used in the academic environment since year 2008 in a capstone course on agile software development using *Scrum.* The *AGIT* model has been used in the industrial environment for the first time in

year 2011. Lessons learned have been included in the current version of the model as described in the Doctoral Dissertation. The *AGIT* model will be further developed in the Laboratory.

The key science contribution of the Doctoral Dissertation is development of comprehensive *AGIT* model for agile software development performance monitoring. The research contributions of the *AGIT* model are:

- definition of key performance indicators;
- definition of metrics for key performance indicators;
- definition of data gathering procedures;
- definition of data organization and
- definition of reporting.

The *AGIT* model has been developed following the PPS concept. Firstly the software development process using Scrum framework has been described. Then the stakeholders have been determined: Management, *Product Owner, Scrum Master and Development Team*. For each stakeholder the goals were defined and for each of the goals key performance indicators have been determined. After that the detailed description of key performance indicators have been presented followed by target and limit values for each key performance indicator. Then the data gathering procedures were defined in compliance with agile principles and the data model was presented. In the end the reporting was introduced.

The feasibility of the *AGIT* model was tested using the case study. The basic information about the company and the project studied is presented as well as the subset of the *AGIT* model key performance indicators used for testing and their graphical presentations: *Velocity diagram, Release Burndown diagram, Iteration Burndown diagram, SPI diagram, CPI diagram* and *BRE spreadsheets*. The case study is presented from the time, organizational and quality aspect. *Release* and *Sprint* flow are presented followed by the project results. Then the measurement results are presented on the *Release* level and *Sprint* level, followed by the results of accuracy analysis of *User Story* points estimation. The results of case study confirmed that the *AGIT* model is compliant with the agile principles regarding data collection and data reporting.

In the conclusion of Doctoral Dissertation the suggestions for further research are given, based on the results of the case study.

Keywords:
Agile software development, AGIT, CMMI, COBIT, Performance indicators, Performance measurement, Planning poker, Product Backlog, Project reporting, PMBOK, Release, Scrum, Performance monitoring, Sprint, Sprint Backlog, SWEBOK, Software development performance, User story

I Z J A V A O A V T O R S T V U

doktorske disertacije

Spodaj podpisana Nataša Žabkar,

z vpisno številko 24004152,

sem avtorica doktorske disertacije z naslovom

MODEL ZA SPREMLJANJE UČINKOVITOSTI
AGILNEGA RAZVOJA PROGRAMSKE OPREME

S svojim podpisom zagotavljam, da:

- sem doktorsko disertacijo izdelala samostojno pod vodstvom mentorja, izrednega profesorja dr. Viljana Mahniča
- so elektronska oblika doktorske disertacije, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko doktorske disertacije
- in soglašam z javno objavo elektronske oblike doktorske disertacije v zbirki »Dela FRI«.

V Ljubljani, dne 22.4.2013 Podpis avtorice:

ZAHVALA

Zahvaljujem se vsem, ki so prispevali k nastanku doktorske disertacije, še posebej izrednemu profesorju dr. Viljanu Mahniču za mentorstvo ter družini in prijateljem za vso podporo in razumevanje v času izdelave tega dela.

KAZALO

1. UVOD 1

1.1 Pregled vsebine doktorske disertacije 1

1.2 Metodološki pristop 1

1.3 Prispevki znanosti 2

1.4 Predpostavke in omejitve doktorske disertacije 2

1.5 Sestava doktorske disertacije 3

2. AGILNI RAZVOJ PROGRAMSKE OPREME 4

2.1 Razvoj programske opreme 4

2.2 Projektno vodenje 11

2.3 Agilni razvoj programske opreme 16

2.4 Metodologija *Scrum* 23

3. MERJENJE UČINKOVITOSTI AGILNEGA RAZVOJA PROGRAMSKE OPREME 29

3.1 Raziskave agilnih metod 29

3.2 Zrelost agilnih metod 30

3.3 Določitev metrik 36

3.4 Poročanje o napredku 42

3.5 Modeli za spremljanje učinkovitosti 44

4. MODEL ZA SPREMLJANJE UČINKOVITOSTI AGILNEGA RAZVOJA PROGRAMSKE OPREME (*AGIT*) 50

4.1 Razvoj modela *AGIT* 50

4.2 Opis modela *AGIT* 52

4.3 Podroben opis kazalnikov 57

4.4 Zbiranje in prikazovanje podatkov 67

5. ŠTUDIJ PRIMERA UPORABE MODELA *AGIT* 76

5.1 Uvod 76

5.2 Opis primera 79

5.3 Spremljanje učinkovitosti *Izdaje* 85

5.4 Spremljanje učinkovitosti prve *Iteracije* 87

5.5 Spremljanje učinkovitosti druge *Iteracije* 90

5.6 Spremljanje učinkovitosti tretje *Iteracije* 92

5.7 Spremljanje učinkovitosti četrte *Iteracije* 94
5.8 Spremljanje učinkovitosti pete *Iteracije* 96
5.9 Spremljanje učinkovitosti šeste *Iteracije* 98
5.10 Spremljanje učinkovitosti sedme *Iteracije* 100
5.11 Točnost ocenjevanja 102
6. ZAKLJUČEK 104
6.1 Povzetek doktorske disertacije 104
6.2 Pregled prispevkov znanosti 104
6.3 Smernice za nadaljnje raziskave 105

Pojmovnik 106
Literatura 108

Seznam slik

Slika 1: *Metodologija projektnega vodenja*
Slika 2: *Procesi v življenjskem ciklu programske opreme (ISO/IEC 12207: 1995)*
Slika 3: *Procesi v življenjskem ciklu programske opreme (ISO/IEC 12207: 2008)*
Slika 4: *COBIT domene (COBIT 4.1)*
Slika 5: *Modeli življenjskega cikla projektov*
Slika 6: *Potek življenjskega cikla projekta*
Slika 7: *Procesne skupine projektnega vodenja (PMBOK)*
Slika 8: *Štiri temeljne vrednote agilnih metod*
Slika 9: *»Agilni dežnik«*
Slika 10: *Primernost uporabe »napovedovalne« ali »prilagojevalne« metode*
Slika 11: *Zaporedni in celoviti pristop k razvoju novega produkta (programske opreme)*
Slika 12: *Postopek razvoja programske opreme z uporabo metodologije Scrum*
Slika 13: *Opis metodologije Scrum na način 3-6-4*
Slika 14: *»Pujsi« in »kure«*
Slika 15: *»Železni« trikotnik*
Slika 16: *Pristopi k merjenju učinkovitosti*
Slika 17: *Deležniki procesa razvoja programske opreme z uporabo metodologije Scrum*
Slika 18: *Struktura modela AGIT*
Slika 19: *Model AGIT*
Slika 20: *Podatkovni model*
Slika 21: *Zgled AGIT nadzorne plošče za prvo Iteracijo*
Slika 22: *Spremembe Hitrosti*
Slika 23: *Padajoči diagram spremljanja napredka Izdaje*
Slika 24: *Padajoči diagram spremljanja napredka prve Iteracije*
Slika 25: *SPI diagram za prvo Iteracijo*
Slika 26: *CPI diagram za prvo Iteracijo*
Slika 27: *Padajoči diagram spremljanja napredka druge Iteracije*
Slika 28: *SPI diagram za drugo Iteracijo*
Slika 29: *CPI diagram za drugo Iteracijo*
Slika 30: *Padajoči diagram spremljanja napredka tretje Iteracije*
Slika 31: *SPI diagram za tretjo Iteracijo*

Slika 32: *CPI diagram za tretjo Iteracijo*
Slika 33: *Padajoči diagram spremljanja napredka četrte Iteracije*
Slika 34: *SPI diagram za četrto Iteracijo*
Slika 35: *CPI diagram za četrto Iteracijo*
Slika 36: *Padajoči diagram spremljanja napredka pete Iteracije*
Slika 37: *SPI diagram za peto Iteracijo*
Slika 38: *CPI diagram za peto Iteracijo*
Slika 39: *Padajoči diagram spremljanja napredka šeste Iteracije*
Slika 40: *SPI diagram za šesto Iteracijo*
Slika 41: *CPI diagram za šesto Iteracijo*
Slika 42: *Padajoči diagram spremljanja napredka sedme Iteracije*
Slika 43: *SPI diagram za sedmo Iteracijo*
Slika 44: *CPI diagram za sedmo Iteracijo*

Seznam tabel

Tabela 1: *Področja znanja programskega inženirstva (SWEBOK 2004)*
Tabela 2: *Metode programskega inženirstva (SWEBOK V3)*
Tabela 3: *Procesna področja CMMI-DEV 1.3*
Tabela 4: *Ključna področja razvoja programske opreme (COBIT 4.1)*
Tabela 5: *Struktura standarda PMBOK Vodnik*
Tabela 6: *Primerjava med COBIT procesi in PMBOK*
Tabela 7: *Ključna področja projektnega vodenja (COBIT 4.1)*
Tabela 8: *Principi agilne programske opreme*
Tabela 9: *Tradicionalni in agilni pristop pri projektih razvoja programske opreme*
Tabela 10: *Scrum vloge*
Tabela 11: *Agilne zrelostne stopnje (AMM)*
Tabela 12: *Scrum preizkus – Scrumbut, Scrumbutt, Nokia preizkus*
Tabela 13: *Agilne metrike*
Tabela 14: *Ključne komponente izračuna prislužene vrednosti*
Tabela 15: *Ključne komponente izračuna agilne prislužene vrednosti*
Tabela 16: *PMBok: Procesi spremljanja in nadzora*
Tabela 17: *Posebni cilji in prakse za spremljanje učinkovitosti*
Tabela 18: *Preglednica notranjih in zunanjih atributov pri razvoju programske opreme*
Tabela 19: *Deležniki procesa razvoja programske opreme z uporabo metodologije Scrum*
Tabela 20: *Cilji deležnikov*
Tabela 21: *Matrika kazalnikov in ciljev*
Tabela 22: *Kazalniki za vlogo Poslovodstvo*
Tabela 23: *Ključne komponenta izračuna SPI in CPI na dnevni ravni*
Tabela 24: *Kazalniki za vlogo Produktni vodja*
Tabela 25: *Primer vprašalnika*
Tabela 26: *Kazalniki za vlogo Razvojna skupina*
Tabela 27: *Kazalniki za vlogo Skrbnik metodologije Scrum*
Tabela 28: *Seznam kazalnikov in ciljnih vrednosti*
Tabela 29: *Seznam kazalnikov in osnovnih metrik*
Tabela 30: *Zbiranje osnovnih metrik*
Tabela 31: *Zbiranje ostalih podatkov*

Tabela 32: *Entitete za projekt*
Tabela 33: *Entitete za Izdajo*
Tabela 34: *Entitete za meritve*
Tabela 35: *Prioritete metrik iz vidika projektnega vodenja*
Tabela 36: *Podmnožica metrik modela AGIT*
Tabela 37: *Potek projekta*
Tabela 38: *Ocenjevanje zahtevnosti Uporabniških zgodb*
Tabela 39: *Scrum vloge*
Tabela 40: *Sestava Razvojne skupine*
Tabela 41: *Kriteriji za potrditev Uporabniške zgodbe*
Tabela 42: *Funkcije orodja za projektno vodenje*
Tabela 43: *Potek Iteracije*
Tabela 44: *Točnost ocenjevanja Uporabniških zgodb (Povprečje)*
Tabela 45: *Točnost ocenjevanja Uporabniških zgodb (Načrtovalni poker)*

1. UVOD

1.1 Pregled vsebine doktorske disertacije

1.1.1 Opredelitev problema

Projekti uvedbe informacijskih rešitev so pogosto ključnega pomena za doseganje ciljev organizacije. Za tovrstne projekte je značilna visoka zahtevnost, kratki roki in visok proračun ter visoka stopnja tveganosti [IIA09]. Konec leta 2010 je Mednarodno združenje za kontrolo in revizijo informacijskih sistemov izvedlo anketo, v kateri je skoraj polovica udeležencev ocenila tveganja projektov s komponento informacijske tehnologije kot zelo visoka in jih uvrstila med najbolj pomembne zadeve, s katerimi se bodo ukvarjali v naslednjem obdobju [ISA11]. Pomembnost in tveganost tovrstnih projektov narekuje povečane zahteve po spremljanju njihove učinkovitosti [ITG07a].

Agilni razvoj programske opreme je nastal pri iskanju rešitev problema visoke stopnje tveganja projektov razvoja programske opreme, pri čemer je bil poudarek na hitrem odzivu na spremembe, ki so bile največkrat vzrok prekoračitev dogovorjenega obsega, rokov in stroškov. Raziskave na področju uspešnosti in učinkovitosti agilnega razvoja programske opreme so pokazale, da je agilni razvoj programske opreme prispeval k povečanju uspešnosti oziroma zmanjšanju odmikov od dogovorjenega obsega in rokov, na področju učinkovitosti oziroma zmanjšanju odmikov od dogovorjenih stroškov pa je področje slabo raziskano ([Mah07a], [Bar05], [Har06], [Sul06]).

Opisana problematika sodi na področje tehnologije programske opreme (ang. »software engineering« ali SE), ki je sestavni del računalniških znanosti. Ožje gledano sodi tema doktorske disertacije na področje zagotavljanja kakovosti programske opreme (ang. »software quality assurance« ali SQA) in znotraj tega na področje zagotavljanja kakovosti razvojnega procesa (ang. »process quality assurance«) ter metrik programske opreme (ang. »software metrics«).

1.1.2 Namen in cilji doktorske disertacije

Namen doktorske disertacije je razviti model za spremljanje učinkovitosti agilnega razvoja programske opreme, ki temelji na sprotnem spremljanju ključnih kazalnikov in združuje spoznanja s področja agilnih metod razvoja programske opreme in področja merjenja učinkovitosti razvojnih procesov.

1.2 Metodološki pristop

Pri izdelavi doktorske dispozicije smo uporabili naslednje metode proučevanja:

- teoretično raziskovanje, ki zajema zbiranje, urejanje, primerjavo in interpretacijo obstoječih dognanj,
- splošno raziskovalno metodo spoznavnega procesa, ki zajema analizo in sintezo spoznanj ter

- empirično preverbo z uporabo študije primera.

Z metodo teoretičnega raziskovanja smo analizirali obstoječo literaturo s področja agilnih metod razvoja programske opreme in področja spremljanja učinkovitosti. Na osnovi zbranih podatkov smo podali pregled obstoječih rešitev spremljanja učinkovitosti agilnega razvoja programske opreme.

Z uporabo splošne raziskovalne metode smo najprej določili, kaj želimo meriti, in opredelili kazalnike, ki jih bomo spremljali. Nato smo oblikovali zasnovo modela za spremljanje učinkovitosti agilnega razvoja programske opreme. Potem smo model dopolnili z opredelitvijo načina zbiranja in prikazovanja podatkov. Naše vodilo pri oblikovanju modela je bilo upoštevanje načel agilnosti.

Uporabnost modela, ki smo ga oblikovali z uporabo splošne raziskovalne metode, smo preverili na študijskem primeru. Ob preverbi smo ugotovili primernost načinov zbiranja in prikazovanja podatkov ter skladnost z načeli agilnosti.

1.3 Prispevki k znanosti

Osnovni prispevek doktorske disertacije k znanosti je izdelava celovitega modela za spremljanje učinkovitosti agilnega razvoja programske opreme. Pri oblikovanju modela za spremljanje učinkovitosti agilnega razvoja programske opreme smo ohranili prednosti obstoječih rešitev za spremljanje agilnega razvoja programske opreme in ublažili njihove pomanjkljivosti.

V okviru oblikovanja modela za spremljanje učinkovitosti agilnega razvoja programske opreme so prispevki k znanosti:

- opredelitev ključnih kazalnikov za spremljanje učinkovitosti;
- opredelitev metrik, potrebnih za izračun ključnih kazalnikov;
- opredelitev postopkov za zbiranje podatkov (izmerjenih vrednosti);
- opredelitev organizacije podatkov (podatkovni model);
- opredelitev oblike poročanja (uporaba diagramov).

Navedeno vsebuje povzetek dobre prakse na področju spremljanja učinkovitosti agilnega razvoja programske opreme.

1.4 Predpostavke in omejitve doktorske disertacije

Izraz »učinkovitost« v doktorski disertaciji je prevod angleškega izraza »performance«.

Model *AGIT* temelji na metodologiji *Scrum*, ki je najbolj razširjena agilna metoda. Predpostavka doktorske disertacije je, da je model *AGIT* uporaben tudi pri drugih agilnih metodologijah.

1.5 Sestava doktorske disertacije

Doktorska disertacija je sestavljena iz šestih poglavij.

Prvo, uvodno, poglavje disertacije vsebuje pregled vsebine doktorske disertacije, metodološki pristop, prispevke znanosti, predpostavke in omejitve doktorske disertacije ter sestavo doktorske disertacije.

V drugem poglavju smo predstavili področje agilnega razvoja programske opreme. Opisali smo razvoj programske opreme in projektno vodenje, agilni razvoj programske opreme in metodologijo *Scrum*.

V tretjem poglavju smo predstavili področje spremljanja učinkovitosti agilnega razvoja programske opreme, ki je predmet doktorske disertacije ter rezultate dosedanjih raziskav na obravnavanem področju.

Model za spremljanje učinkovitosti agilnega razvoja programske opreme, ki smo ga poimenovali *AGIT*, smo opisali v četrtem poglavju. Na začetku smo predstavili izbrani pristop k reševanju problema. Potem smo podali splošen opis modela *AGIT* ter način zbiranja in prikazovanja podatkov.

Študij primera uporabe modela *AGIT* smo opisali v petem poglavju in predstavili rezultate merjenja učinkovitosti na ravni *Izdaje* in na ravni *Iteracije*.

Šesto, zaključno, poglavje vključuje povzetek doktorske disertacije, pregled prispevkov znanosti in smernice za nadaljnje raziskave.

2. AGILNI RAZVOJ PROGRAMSKE OPREME

2.1 Razvoj programske opreme

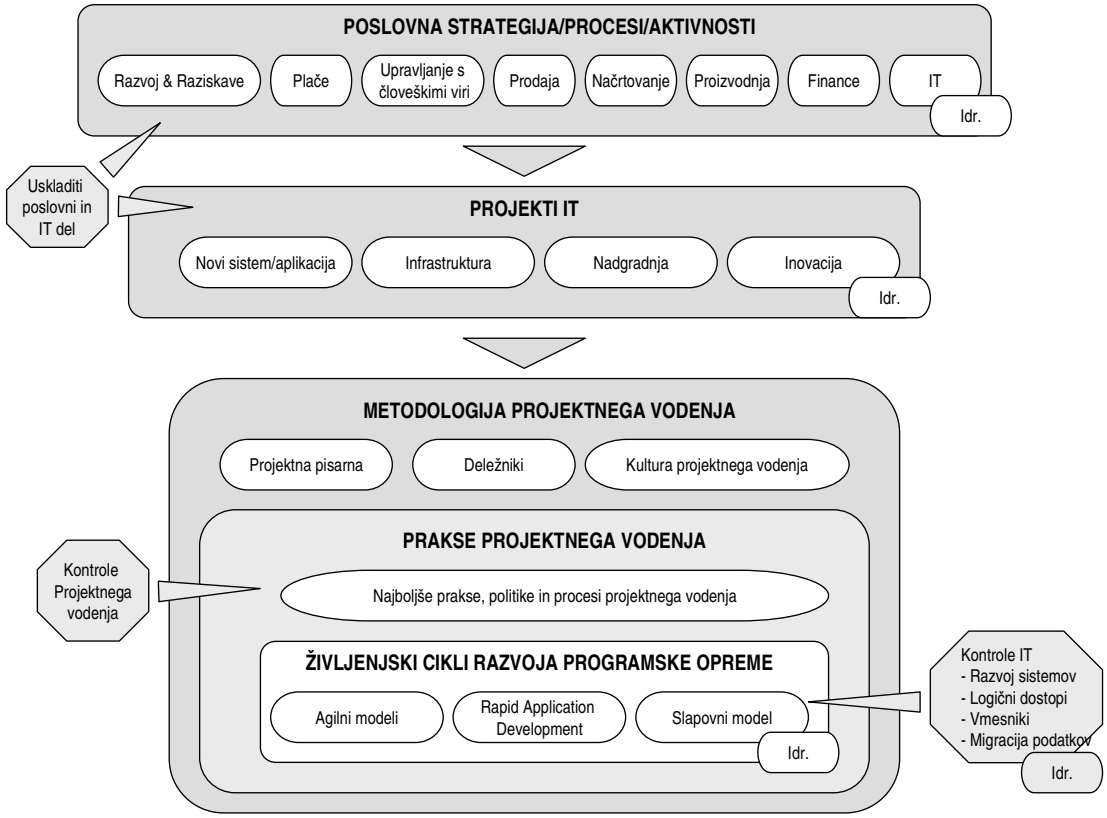
2.1.1 Uvod

Razvoj programske opreme skladno z opredelitvijo v standardu ISO 9000-3 [ISO04] zajema vse aktivnosti, ki jih je potrebno izvesti, da bi ustvarili programski proizvod. Isti standard določa, da je programski proizvod celotni skupek računalniških programov, postopkov ter pripadajoče dokumentacije in podatkov, namenjenih za dobavo uporabniku.

Razvoj programske opreme sodi v znanstveno področje tehnologije programske opreme oziroma programskega inženirstva (ang. software engineering), ki zajema sistematično uporabo znanstvenih in tehnoloških znanj, metod in izkušenj za potrebe načrtovanja, izgradnje, testiranja in dokumentiranja programske opreme [Piv96].

Povezava med razvojem programske opreme in projektnim vodenjem je opisana v smernici za revidiranje projektov informacijske tehnologije »Global Technology Audit Guide (GTAG) 12: Auditing IT Projects« [IIA09] v grafični obliki, kot je predstavljena na Sliki 1.

Slika 1: Metodologija projektnega vodenja



Vir: [IIA09]

Poslovna strategija se uresničuje z izvajanjem projektov informacijske tehnologije, ki potekajo skladno z metodologijo projektnega vodenja, le ta pa vsebuje prakse projektnega vodenja in življenjski cikel razvoja programske opreme. Uporaba metodologije projektnega vodenja prispeva k doseganju uspešnosti in učinkovitosti projekta [IIA09].

2.1.2 Technical Report ISO/IEC TR 19759 (SWEBOK)

»Vodnik po znanju programskega inženirstva SWEBOK« (ang. Software Engineering Body of Knowledge) [IEE04] je rezultat dela komisije za koordinacijo programskega inženirstva (ang. Software Engineering Coordinating Committee), ki deluje pri združenju IEEE Computer Society.

Trenutno je v veljavi njegova različica, ki je izšla konec leta 2004 in je znana tudi kot SWEBOK 2004 ali tehnično poročilo ISO/IEC TR 19759 (ang. Technical Report ISO/IEC TR 19759). V pripravi je različica SWEBOK V3.

SWEBOK 2004 vsebuje 10 področij znanja, ki so navedena v Tabeli 1.

Tabela 1: Področja znanja programskega inženirstva (SWEBOK 2004)

Oznaka	Ključno področje (slovensko)	Ključno področje (angleško)
1	Zahteve programske opreme	Software Requirements
2	Programska zasnova	Software Design
3	Izdelava programske opreme	Software Construction
4	Preskušanje programske opreme	Software Testing
5	Vzdrževanje programske opreme	Software Maintenance
6	Upravljanje konfiguracije programske opreme	Software Configuration Management
7	Upravljanje programskega inženirstva	Software Engineering Management
8	Proces programskega inženirstva	Software Engineering Process
9	Orodja in metode programskega inženirstva	Software Engineering Tools and Methods
10	Kakovost programske opreme	Software Quality

Vir: [IEE04]

Metode programskega inženirstva predstavljajo organiziran in sistematičen pristop k razvoju programske opreme na ciljnem računalniku. Ena izmed možnih kategorizacij metod programskega inženirstva je podana v predlogu nove različice Vodnika po znanju programskega inženirstva (SWEBOK V3) [IEE08] in je prikazana v Tabeli 2.

Tabela 2: Metode programskega inženirstva (SWEBOK V3)

Oznaka	Metode programskega inženirstva (slovensko)	Metode programskega inženirstva (angleško)
1	Heuristične metode	Heuristic Methods
2	Formalne metode	Formal Methods
3	Prototipne metode	Prototyping Methods
4	Agilne metode	Agile Methods

Vir: [IEE08]

2.1.3 Standard ISO/IEC 12207: »Informacijska tehnologija - Procesi življenjskega cikla programske opreme«

Vodnik SWEBOK se velikokrat sklicuje na mednarodni standard ISO/IEC 12207: »Informacijska tehnologija - Procesi življenjskega cikla programske opreme« (ang. Information technology - Software life cycle processes) [ISO95], ki opisuje procese v življenjskem ciklu programske opreme. Osnovni namen standarda ISO/IEC 12207 je zagotoviti skupen okvir in terminologijo za kupce, dobavitelje, razvijalce, vzdrževalce, operaterje, managerje in tehnike, ki so vključeni v razvoj programske opreme in sicer z opredelitvijo procesov.

Standard ISO/IEC 12207 je pripravila skupna tehnična komisija ISO/IEC JTC 1, Informacijska tehnologija, podkomisija SC7, programski in sistemski inženiring (ang. Software Engineering Coordinating Committee) pri organizaciji ISO (International Organization for Standardization) in komisiji IEC (International Electrotechnical Commission). Pri pripravi standarda je sodelovalo tudi združenje IEEE Computer Society. Prva izdaja standarda ISO/IEC 12207 je bila objavljena leta 1995 [ISO95] in potem leta 2002 dopolnjena z amandmajem številka 1 ter leta 2004 z amandmajem številka 2. Zadnja, druga izdaja je bila objavljena leta 2008 [ISO08], z oznako ISO/IEC 12207:2008, IEEE Std 12207-2008: »Sistemski in programski inženiring - Procesi življenjskega cikla programske opreme«.

Standard ISO/IEC 12207:1995 opredeljuje 22 procesov, 95 aktivnosti, 325 opravil in 254 ciljev v življenjskem ciklu programske opreme. Procesi so razvrščeni v tri kategorije, kot je razvidno iz Slike 2:

- primarni procesi (nabavljanje, dobavljanje, razvijanje, delovanje in vzdrževanje),
- podporni procesi (dokumentiranje, upravljanje konfiguracije, zagotavljanje kakovosti, verificiranje, validiranje, skupno pregledovanje in presojanje) ter
- organizacijski procesi življenjskega cikla (vodenje, infrastruktura, izboljševanje in usposabljanje).

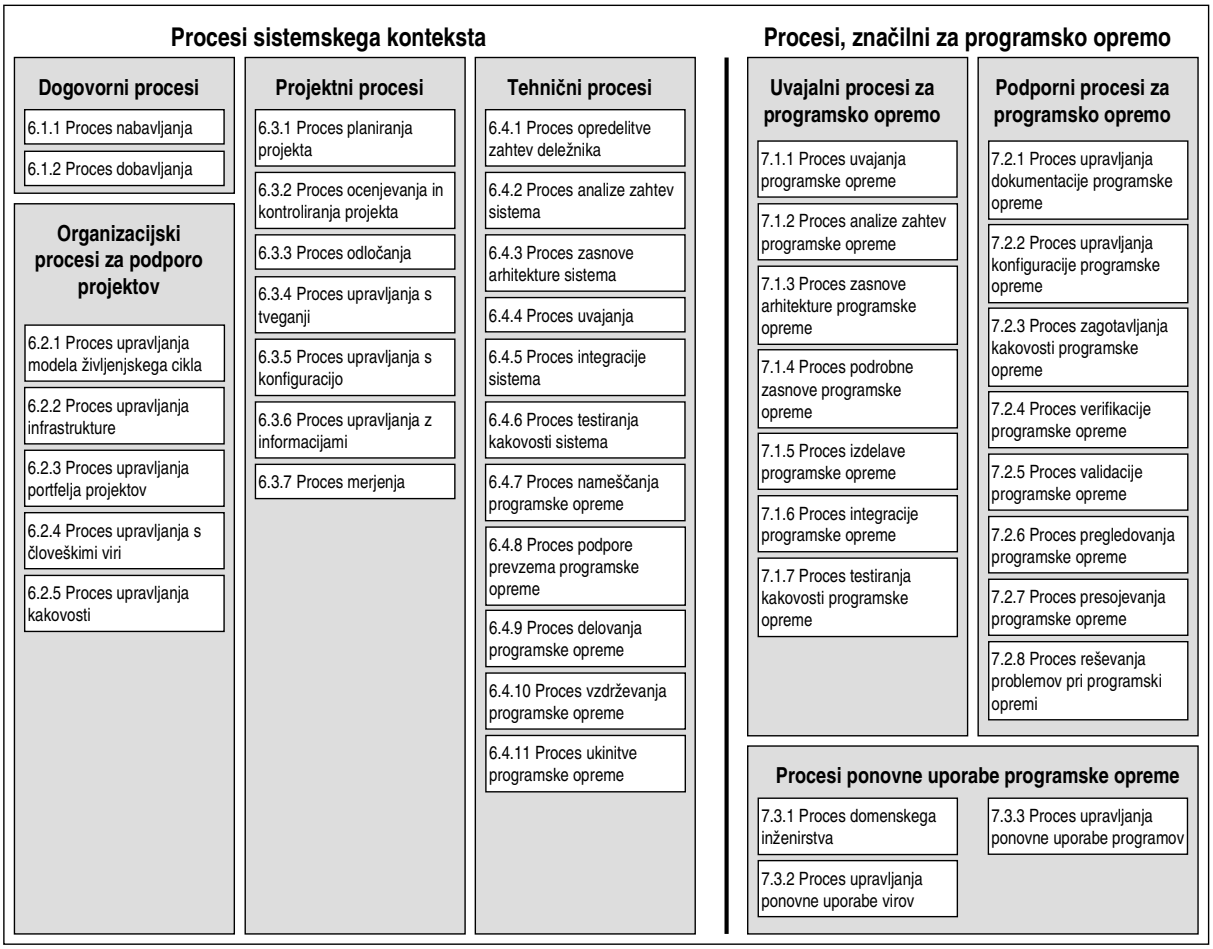
Slika 2: Procesi v življenjskem ciklu programske opreme (ISO/IEC 12207: 1995)

Vir: [ISO95]

V različici iz leta 2008 je opredeljenih 43 procesov. Procesi so razvrščeni v dve glavni kategoriji, kot je razvidno iz Slike 3:

- kategorija procesov sistemskega konteksta, ki ima štiri podkategorije (dogovorni procesi, organizacijski procesi za podporo projektov, projektni procesi in tehnični procesi) ter
- kategorija procesov, značilnih za programsko opremo, ki ima tri podkategorije (uvajalni procesi za programsko opremo, podporni procesi za programsko opremo in procesi ponovne uporabe programske opreme).

Slika 3: Procesi v življenjskem ciklu programske opreme (ISO/IEC 12207: 2008)



Vir: [ISO08]

2.1.4 Združeni zmožnostno zrelostni model (CMMI)

Vodnik SWEBOK se velikokrat sklicuje tudi na model CMMI: »Združeni zmožnostno zrelostni model« (ang. Capability Maturity Model Integration) [CMM06], ki je model za izboljšanje in ocenitev učinkovitosti razvojnih organizacij. Zmožnost procesa je opis niza pričakovanih rezultatov, ki jih lahko dosežemo, če sledimo procesu. Zrelost procesa je obseg opredeljenosti, upravljanosti, merjenosti, nadzorovanosti in učinkovitosti posameznega procesa

Model CMMI je nastal iz modela CMM, ki ga je razvil Inštitut za programsko inženirstvo SEI (ang. Software Engineering Institute) pod sponzorstvom ameriškega ministrstva za obrambo, vlade in v sodelovanju z industrijo. Zadnja različica modela CMMI V1.3 je bila objavljena novembra 2010 [CMM10] in vsebuje model CMMI za razvoj (ang. CMMI for development; CMMI-DEV), model CMMI za storitve (ang. CMMI for services; CMMI-SVC) in model CMMI za nabavo (ang. CMMI for acquisition; CMMI-ACQ).

Model CMMI za razvoj obravnava 22 procesnih področij, ki so razvrščeni v štiri kategorije: upravljanje procesov, projektno vodenje, inženiring in podpora, kot je razvidno iz Tabele 3. Za posamezno procesno področje je možno določiti raven zmožnosti (od najnižje ravni 0 do

najvišje ravni 3), za več procesnih področij pa raven zrelosti (od najnižje ravni 1 do najvišje ravni 5).

Tabela 3: Procesna področja CMMI-DEV 1.3

ZŠ	Procesno področje (angleščina)	Procesno področje (slovenščina)	Kategorija	Zrelostna raven
1	CAR-Causal Analysis and Resolution	CAR-Analiza vzrokov in rešitev	Podpora	5
2	CM- Configuration Management	CM-Upravljanje konfiguracije	Podpora	2
3	DAR-Decision Analysis and Resolution	DAR-Odločitvena analiza in rešitev	Podpora	3
4	IPM-Integrated Project Management	IPM-Integrirano vodenje projektov	Projektno vodenje	3
5	MA-Measurement and Analysis	MA-Merjenje in analiza	Podpora	2
6	OPD-Organizational Process Definition	OPD-Opredelitev procesov organizacije	Upravljanje procesov	3
7	OPF-Organizational Process Focus	OPF-Procesna usmeritev organizacije	Upravljanje procesov	3
8	OPM-Organizational Performance Management	OPM-Upravljanje učinkovitosti organizacije	Upravljanje procesov	5
9	OPP-Organizational Process Performance	OPP-Učinkovitost procesov v organizaciji	Upravljanje procesov	4
10	OT-Organizational Training	OT-Usposabljanje v organizaciji	Upravljanje procesov	3
11	PI-Product Integration	PI-Integracija izdelka	Inženiring	3
12	PMC-Project Monitoring and Control	PMC-Spremljanje in nadzor projekta	Projektno vodenje	2
13	PP-Project Planning	PP-Planiranje projektov	Projektno vodenje	2
14	PPQA-Process and Product Quality Assurance	PPQA-Zagotavljanje kakovosti procesov in proizvodov	Podpora	2
15	QPM-Quantitative Project Management	QPM-Kvantitativno projektno vodenje	Projektno vodenje	4
16	RD-Requirements Development	RD-Razvoj zahtev	Inženiring	3
17	REQM-Requirements Management	REQM-Upravljanje zahtev	Projektno vodenje	2
18	RSKM-Risk Management	RSKM-Upravljanje tveganj	Projektno vodenje	3
19	SAM-Supplier Agreement Management	SAM-Upravljanje pogodb z dobavitelji	Projektno vodenje	2
20	TS-Technical Solution	TS-Tehnične rešitve	Inženiring	3
21	VAL-Validation	VAL-Validacija	Inženiring	3
22	VER-Verification	VER-Verifikacija	Inženiring	3

Vir: [CMM10]

Model CMMI-DEV V1.3 [CMM10] podpira agilni razvoj programske opreme in vsebuje opombe za uporabnike agilnih metod za procesna področja: CM, PI, PMC, PP, PPQA, RD, REQM, RSKM, TS, in VER. Uporaba modela CMMI v agilnem okolju je podrobneje opisana v [Gla08].

V modelu CMMI-DEV V1.3 so za vsak proces navedeni generični cilji (ang. »generic goal, oznaka GG) in generični postopki (ang. »generic practice«, oznaka GP), ter posebni cilji (oznaka SG, ang. »specific goal«) in posebni postopki (oznaka SP, ang. »specific practice«). Generični in posebni cilji so zahtevane komponente, kar pomeni, da so nujni za izboljšanje procesa. Generični cilji so skupni vsem procesnim področjem, posebni cilji pa so značilni za posamezno procesno področje. Generični in posebni cilji opisujejo značilnosti, ki morajo biti dosežene, da bi se proces uveljavil oziroma izpolnil zahteve glede ravni zmožnosti in/ali zrelosti. Generične in posebne prakse so pričakovane komponente, kar pomeni, da so pomembne pri doseganju z njimi povezanih ciljev.

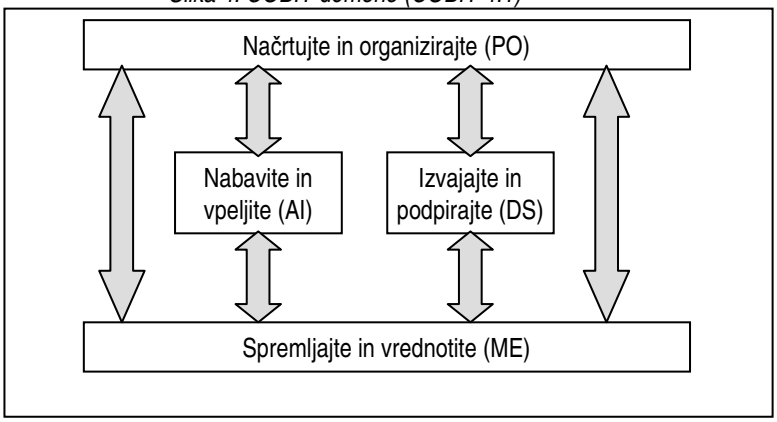
2.1.5 Ogrodje COBIT in razvoj programske opreme

Model CMMI-DEV V1.3 se sklicuje na ogrodje COBIT (ang. »Control Objectives for Information and related Technology«) [ITG07a] za upravljanje informacijske tehnologije. Ogrodje COBIT vsebuje dobro prakso na celotnem področju uporabe informacijske tehnologije. Uporaba COBIT-a je priporočena v Mednarodnih standardih za strokovnjake revidiranja, kontrol in dajanja zagotovil na področju informacijske tehnologije.

Zadnja različica ogrodja COBIT z oznako COBIT 5 je bila objavljena aprila 2012.

Na najvišji ravni ima ogrodje COBIT štiri domene, prikazane na Sliki 4. Vsaka domena ima večje število procesov, vsak izmed procesov pa ima več kontrolnih ciljev. Prva domena »Načrtujte in organizirajte« (PO) zajema strateške procese na področju informacijske tehnologije. Druga domena »Nabavite in vpeljite (AI)« se ukvarja z uresničevanjem strateških ciljev skozi razvoj ali nabavo ter uvajanje informacijske tehnologije. Tretja domena »Izvajajte in podpirajte (DS)« pokriva izvajanje poslovnih procesov, podprtih z uporabo informacijske tehnologije. Izhodi iz vseh naštetih domen so vhodi v četrto domeno »Spremljajte in vrednotite (ME)«, ki vključuje nadzorni vidik upravljanja informacijske tehnologije.

Slika 4: COBIT domene (COBIT 4.1)



Vir: [ITG07a]

V Tabeli 4 je podan izbor ključnih področij razvoja programske opreme v ogrodju COBIT.

Tabela 4: Ključna področja razvoja programske opreme (COBIT 4.1)

Oznaka	Kontrolni cilj (COBIT)
	Specificiranje zahtev
AI1.1	Opredelitev in vzdrževanje poslovnih funkcionalnih in tehničnih zahtev
AI1.2	Poročilo o analizi tveganja
AI1.3	Studija izvedljivosti in oblikovanje alternativnih možnosti
AI1.4	Odločitev in odobritev zahtev in izvedljivosti
	Načrtovanje aplikacije
AI2.1	Visoko nivojska zasnova
AI2.2	Podrobna zasnova
AI2.3	Aplikacijske kontrole in primernost za revidiranje
AI2.4	Varnost aplikacije in razpoložljivost
AI5.1	Kontrola nabave
AI3.1	Načrt za nabavo tehnološke infrastrukture
AI5.4	Nabava virov IT
	Izdelava aplikacije
AI2.7	Razvoj aplikacijske programske opreme

Oznaka	Kontrolni cilj (COBIT)
AI2.8	Zagotovitev kakovosti programske opreme
AI4.1	Načrtovanje operativnih rešitev
AI4.2	Prenos znanja poslovnemu vodstvu
AI4.3	Prenos znanja končnim uporabnikom
AI4.4	Prenos znanja operativnemu in podpornemu osebju
AI7.1	Usposabljanje
	Testiranje aplikacije
AI7.2	Načrt testiranja
AI7.3	Načrt vpeljave
AI7.4	Testno okolje
AI7.5	Pretvorba sistema in podatkov
AI7.7	Končni sprejemni test
	Uvedba aplikacije v produkcijo
AI7.8	Prenos v produkcijo
AI7.9	Pregled po vpeljavi

Vir: [ITG07a]

2.2 Projektno vodenje

2.2.1 Uvod

Proces izdelave programske opreme je inženirski proces in projektni proces [Piv96]. V predhodnem poglavju smo predstavili izbor metod iz področja programskega inženiringa, v tem poglavju pa bomo podali izbor metod iz področja projektnega vodenja.

V »Vodniku po znanju projektnega vodenja PMBOK« [PMI08b] je projekt opredeljen kot začasno prizadevanje za uresničitev edinstvenega izdelka, storitve ali rezultata. Začasno prizadevanje pomeni, da ima projekt določen začetek in konec ter da ni del tekočega oz. operativnega poslovanja. Edinstvenost se nanaša na posebnost izdelka, zaradi katere njegova uresničitev ni del tekočega poslovanja. Projektno vodenje je opredeljeno kot uporaba znanja, veščin, tehnik in orodij v aktivnostih projekta za izpolnitev njegovih zahtev; aktivnost projekta pa je opredeljena kot element dela, ki se izvaja med potekom projekta in ima praviloma: pričakovano trajanje, pričakovane stroške in pričakovane potrebe po virih.

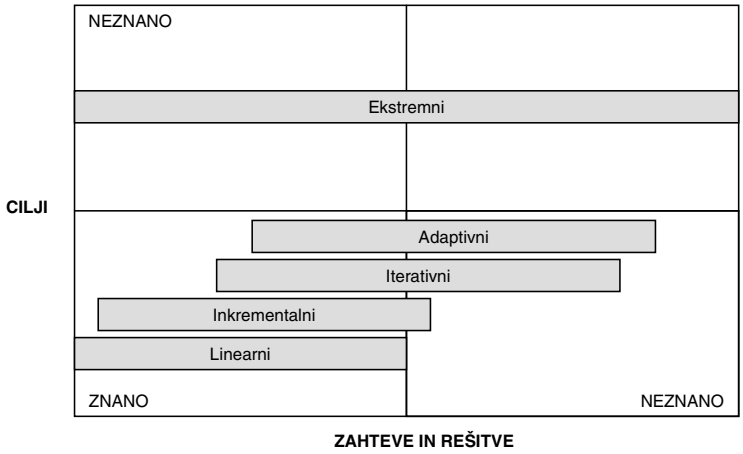
Ena izmed možnih kategorizacij model življenjskega cikla projektov je predstavljena na sliki 25 [Wys09], pri čemer so osnovne kategorije:

- TPM: tradicionalni modeli (linearni in inkrementalni);
- APM: agilni modeli (iterativni in adaptivni) ter
- xPM: ekstremni modeli.

Na Sliki 5 je podan vpliv negotovosti glede ciljev in zahtev ter rešitev projekta na izbiro modela. Tradicionalni modeli so primerni za projekt, pri katerem so cilji, zahteve in rešitev znani, za ostale primera pa so bolj primerni agilni ter ekstremni modeli.

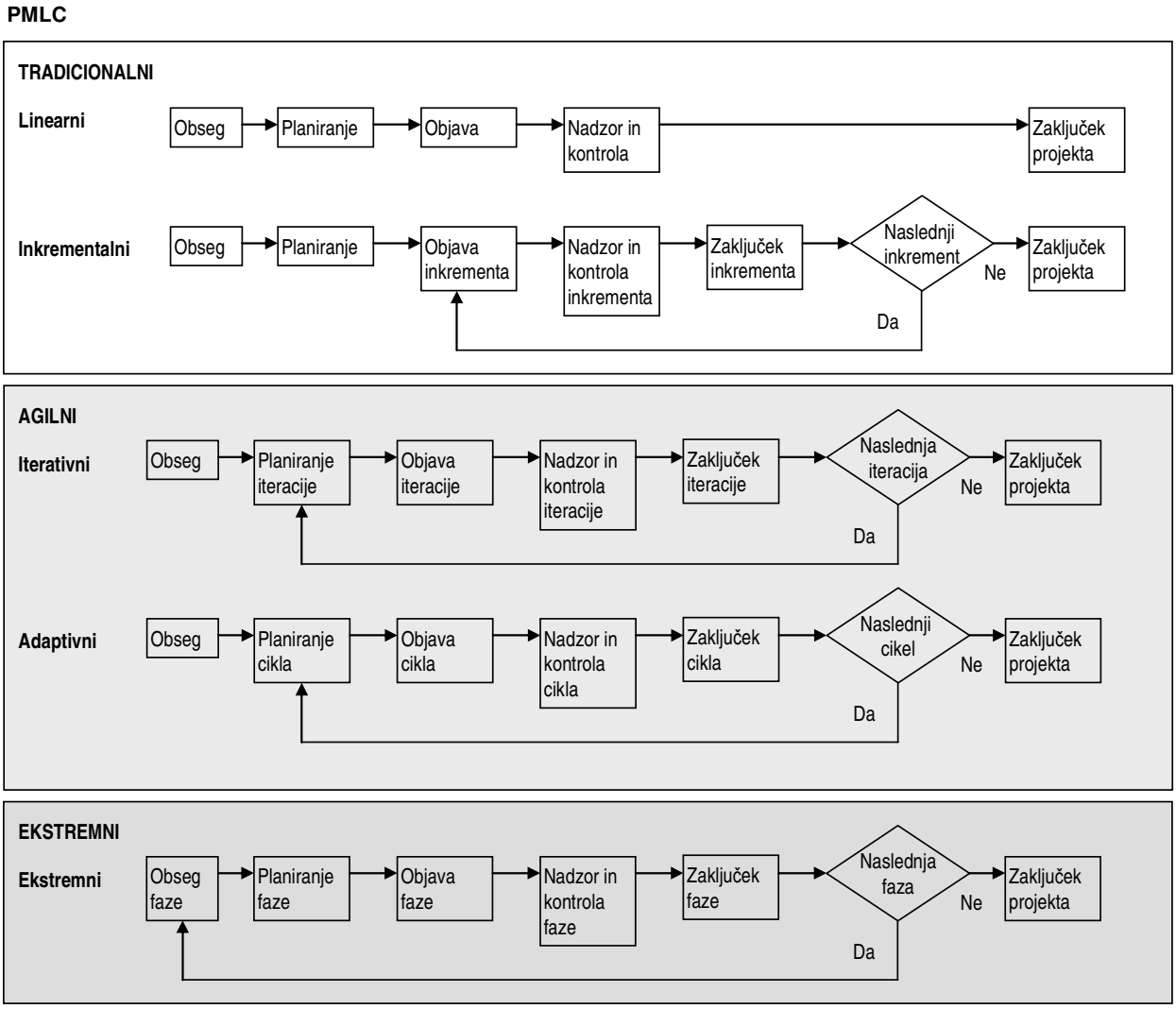
Modeli, predstavljeni na Sliki 5 vsebujejo naslednje faze: obseg, planiranje, objavo, nadzor in kontrolo ter zaključek projekta. Pri vseh modelih je začetna aktivnost opredelitev obsega, zadnja pa zaključek projekta.

Slika 5: Modeli življenjskega cikla projektov



Modeli se razlikujejo glede možnosti vrnitve v predhodne faze ter glede točke, v katero se vrnejo po zaključku posameznega dela projekta (inkrement, iteracija, cikel, faza), kar je razvidno iz Slike 6.

Slika 6: Potek življenjskega cikla projekta



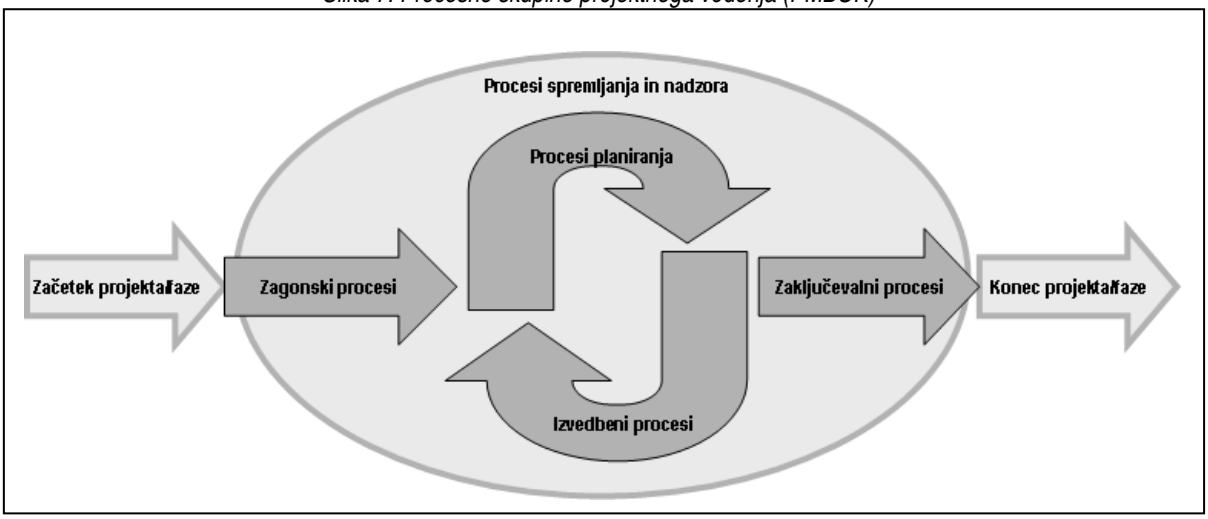
Vir: [Wys09]

2.2.2 Standard ANSI/PMI 99-001-2008 (PMBOK Vodnik)

Standard PMBOK Vodnik ali »Vodnik po znanju projektnega vodenja« (ang. Guide to the Project Management Body of Knowledge) [PMI08a] je razvil Inštitut za projektno vodenje (ang. Project Management Institute ali PMI) in je zelo razširjen v Severni Ameriki. Trenutno je v veljavi njegova četrta izdaja, ki je izšla konec leta 2008 in je priznana kot ameriški nacionalni standard ANSI/PMI 99-001-2008. PMBOK Vodnik se nanaša na vse projekte in ni posebej prilagojen projektom s komponento informacijske tehnologije.

PMBOK Vodnik [PMI08a] ima 42 procesov, razporejenih v petih procesnih skupin projektnega vodenja in devet področij znanja. Pet procesnih skupin odraža faze življenjskega cikla projekta in so predstavljene na Sliki 7 in Tabeli 5.

Slika 7: Procesne skupine projektnega vodenja (PMBOK)



Vir: ([PMI08a])

Devet področij znanja vključujejo znanja, potrebna za kvalitetno vodenje projekta in so navedena v Tabeli 5.

Tabela 5: Struktura standarda PMBOK Vodnik

Področja znanja	**Procesne skupine**				
Procesi	**Zagonski procesi**	**Načrtovalni procesi**	**Izvedbeni procesi**	**Procesi spremljanja in nadzora**	**Zaključevalni procesi**
Področje znanja: 4. Obvladovanje integracije					
4.1 Razvoj projektne listine	X				
4.2 Razvoj načrta projektnega vodenja		X			
4.3 Usmerjanje in vodenje izvedbe projekta			X		
4.4 Nadzor dela				X	
4.5 Integracija nadzora nad spremembami				X	
4.6 Zaključek projekta ali faze					X
Področje znanja: 5. Obvladovanje obsega					
5.1 Načrtovanje obsega		X			
5.2 Opredelitev obsega		X			

Področja znanja	Procesne skupine				
Procesi	**Zagonski procesi**	**Načrtovalni procesi**	**Izvedbeni procesi**	**Procesi spremljanja in nadzora**	**Zaključevalni procesi**
5.3 Razdelitev projekta na manjše dele		X			
5.4 Potrditev obsega				X	
5.5 Nadzor obsega				X	
Področje znanja: 6. Obvladovanje časa					
6.1 Opredelitev aktivnosti		X			
6.2 Razporeditev aktivnosti		X			
6.3 Ocena potrebnih virov		X			
6.4 Ocena potrebnega časa		X			
6.5 Razvoj urnika aktivnosti		X			
6.6 Nadzor nad urnikom aktivnosti				X	
Področje znanja: 7. Obvladovanje stroškov					
7.1 Ocena stroškov		X			
7.2 Načrt stroškov		X			
7.3 Nadzor stroškov				X	
Področje znanja: 8. Obvladovanje kakovosti					
8.1 Načrtovanje kakovosti projekta		X			
8.2 Zagotavljanje kakovosti			X		
8.3 Nadzor kakovosti				X	
Področje znanja: 9. Obvladovanje človeških virov					
9.1 Načrtovanje človeških virov		X			
9.2 Organizacija ekipe			X		
9.3 Razvoj projektne ekipe			X		
9.4 Upravljanje projektne ekipe			X		
Področje znanja: 10. Obvladovanje komuniciranja					
10.1 Opredelitev deležnikov	X				
10.2 Načrtovanje komunikacij		X			
10.3 Razširjanje informacij			X		
10.4 Upravljanje s pričakovanji deležnikov			X		
10.5 Poročanje o doseganju ciljev				X	
Področje znanja: 11. U Obvladovanje tveganj					
11.1 Načrtovanje upravljanja tveganj		X			
11.2 Opredelitev tveganj		X			
11.3 Kvalitativna analiza tveganj		X			
11.4 Kvantitativna analiza tveganj		X			
11.5 Načrtovanje odziva na tveganja		X			
11.6 Spremljanje in nadzor				X	
Področje znanja: 12. Obvladovanje oskrbovanja					
12.1 Načrtovanje oskrbovanja		X			
12.2 Izvedba oskrbovanja			X		
12.3 Upravljanje oskrbovanja				X	
12.4 Zaključek oskrbovanja					X

Vir: po [PMI08b]

Agilni vidik PMBOK je razviden iz vsebine izpita za pridobitev naziva »PM Agile Certified Practitioner (PMI-ACP)«:

- Področje I: Dobava glede na vrednost (ang. value-driven delivery);
- Področje II: Vključenost deležnikov (ang. stakeholder engagement);
- Področje III: Izboljševanje praks za učinkovitost skupine (ang. boosting team performance practices);

- Področje IV: Prilagodljivo planiranje (ang. adaptive planning);
- Področje V: Odkrivanje in reševanje problemov (ang. problem detection and resolution) in
- Področje VI: Stalno izboljševanje: izdelka, procesov in oseb (ang. continuous improvement: product, process, people).

Opisi vsebine izpita, ki se nanašajo na merjenje učinkovitosti vključujejo:
- planiranje, nadzor in prilagajanje, ki vključuje vsaj: retrospektivne sestanke, plošče z nalogami/Kanban plošče, časovne okvirje, načrtovanje iteracije in izdaje, omejitve nedokočanega dela, rastoči/padajoči diagram spremljanja napredka, kumulativni diagrami toka, prilagajanje procesa;
- metrike, ki vključujejo vsaj: hitrost, čas cikla, prisluženo vrednost za agilne projekte, spregledane napake (ang. escaped defects)

2.2.3 Ogrodje COBIT in projektno vodenje

Ogrodje COBIT [ITG07a] vsebuje dobro prakso na celotnem področju uporabe informacijske tehnologije, vključno z upravljanjem projektov in med viri navaja tudi predhodno opisan standard PMBOK.

V Tabeli 6 je navedena primerjava med COBIT procesi in standardom PMBOK.

Tabela 6: Primerjava med COBIT procesi in PMBOK

COBIT: Domena IT/ Proces IT	PMBOK
PO-Načrtujte in organizirajte	
PO1-Opredelite strateški načrt za IT	
PO2-Opredelite informacijsko arhitekturo	
PO3-Določite tehnološko usmeritev	
PO4-Opredelite procese, organizacijo in razmerja IT	
PO5-Upravljajte investicije v IT	
PO6-Sporočajte cilje in usmeritev vodstva	
PO7-Upravljajte človeške vire v sektorju IT	
PO8-Upravljajte kakovost	Da
PO9-Upravljajte in obvladujte tveganja IT	
PO10-Upravljajte projekte	Da
AI-Nabavite in vpeljite	
AI1-Določite avtomatizirane rešitve	Da
AI2-Nabavite in vzdržujte aplikacijske programe	Da
AI3-Nabavite in vzdržujte tehnološko infrastrukturo	Da
AI4-Omogočite delovanje in uporabo	Da
AI5-Zagotovite vire IT	
AI6-Upravljajte spremembe	Da
AI7-Namestite in potrdite rešitve in spremembe	Da
DS-Izvajajte in podpirajte	
DS1-Opredelite in upravljajte ravni storitve	Da
DS2-Upravljajte storitve tretje stranke	Da
DS3-Upravljajte delovanje in zmogljivost	Da
DS4-Zagotovite neprekinjenost storitev	Da
DS5-Zagotovite varnost sistemov	Da
DS6-Ugotovite in porazdelite stroške	
DS7-Izobrazite in usposobite uporabnike	Da
DS8-Upravljajte službo za pomoč uporabnikom in obvladujte incidente	

COBIT: Domena IT/ Proces IT	PMBOK
DS9-Upravljajte konfiguracijo	
DS10-Upravljajte probleme	
DS11-Upravljajte podatke	
DS12-Upravljajte fizično okolje	
DS13-Upravljajte delovanje	
ME-Spremljajte in vrednotite	
ME1-Spremljajte in vrednotite delovanje IT	
ME2-Spremljajte in vrednotite notranje kontrole	
ME3-Zagotovite skladnost z zunanjimi zahtevami	Da
ME4-Zagotovite upravljanje IT	

Vir: [ITG07a]

Kontrolni cilji COBIT procesa »Upravljanje projektov« (PO10) so navedeni v Tabeli 7.

Tabela 7: Ključna področja projektnega vodenja (COBIT 4.1)

Oznaka	Kontrolni cilj (COBIT)
PO10.1	Ogrodje za upravljanje programov
PO10.2	Ogrodje za upravljanje projektov
PO10.3	Pristop k upravljanju projekta
P010.4	Zavezanost udeležencev
PO10.5	Izjava o obsegu projekta
PO10.6	Začetek faze projekta
P010.7	Integrirani projektni načrt
PO10.8	Viri projekta
PO10.9	Obvladovanje tveganj pri projektu
PO10.10	Načrt kakovosti projekta
PO10.11	Nadzor sprememb projekta
PO10.12	Projektno načrtovanje jamstvenih metod
PO10.13	Merjenje izvedbe projekta, poročanje in spremljanje
PO10.14	Zaključek projekta

Vir: [ITG07a]

2.3 Agilni razvoj programske opreme

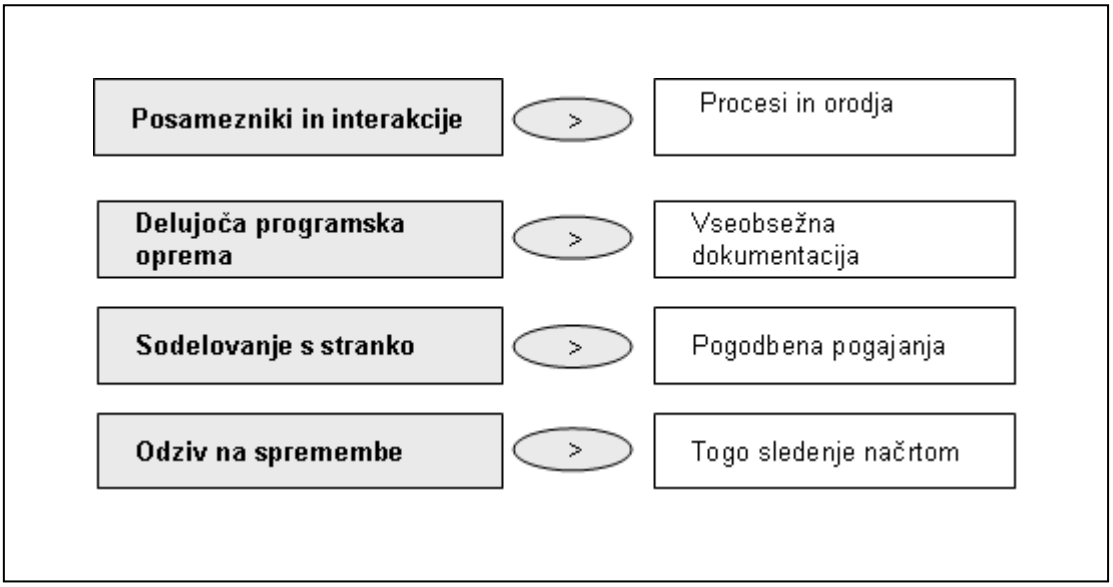
2.3.1 Uvod

V Wikipediji [Wik12a] je navedeno, da agilni razvoj programske opreme zajema skupino (družino) metod za razvoj programske opreme, ki temelji na iterativnem razvoju, pri katerem razvoj zahtev in rešitev poteka v okviru sodelovanja med samoorganiziranimi interdisciplinarnimi skupinami. Večje število agilnih metod je nastalo že prej[1], vendar pa se je skupni izraz »agilne metode« začel uporabljati šele po decembru leta 2001, ko je bil objavljen Manifest agilnega razvoja programske opreme [Bec01a], po sestanku 17 strokovnjakov na področju agilnega razvoja.

[1] Wikipedija navaja, da je praksa testiranja pred mikro-napredkom uporabila že NASA v zgodnjih 1960-tih pri projektu Mercury (Larman 2003), praksa refactoringa, pristopa od spodaj navzgor in postopnega načrtovanja pa opisal Leo Brodie leta 1984 v knjigi »Thinking Forth«.

Agilni razvoj je razvoj programske opreme, ki temelji na vrednotah, navedenem v Agilnem manifestu [Nic09]. Na sliki 8 so prikazane temeljne vrednote agilnikov, pri čemer velja, da agilniki spoštujejo vrednote na levi in desni, vendar dajejo prednost vrednotam na levi.

Slika 8: Štiri temeljne vrednote agilnih metod



Vir: [Bec01a]

V Tabeli 8 je navedenih dvanajst principov v ozadju agilne programske opreme.

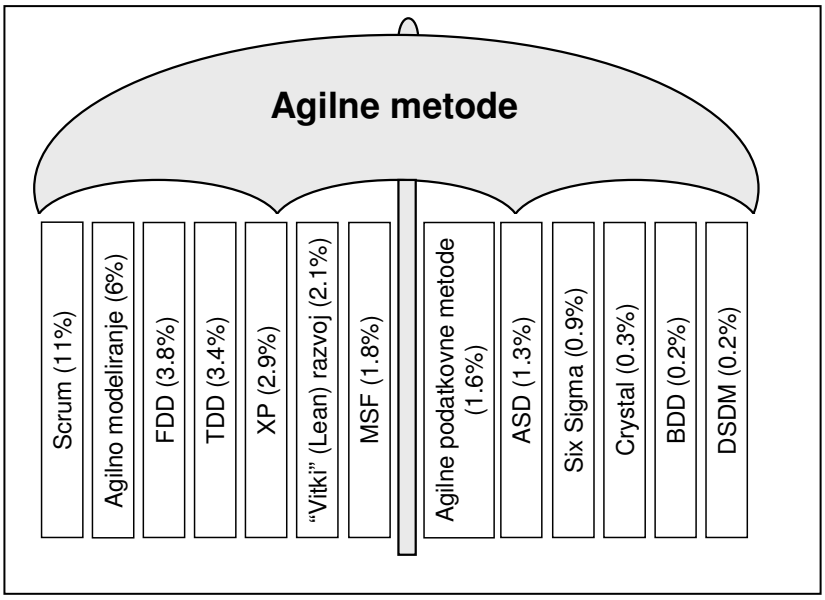
Tabela 8: Principi v ozadju agilnega manifesta

ZŠ	Opis
1	Naša najvišja prioriteta je zadovoljiti stranko z zgodnjim in nepretrganim izdajanjem vredne programske opreme.
2	Sprejemamo spremembe zahtev, celo v poznih fazah razvoja. Agilni procesi vprežejo tovrstne spremembe v prid konkurenčnosti naše stranke.
3	Delujočo programsko opremo izdajamo pogosto, znotraj obdobja nekaj tednov, do nekaj mesecev, s preferenco po krajšem časovnem okvirju.
4	Poslovneži in razvijalci morajo skozi celoten projekt dnevno sodelovati.
5	Projekte gradimo okrog motiviranih posameznikov. Omogočimo jim delovno okolje, nudimo podporo in jim zaupamo, da bodo svoje delo opravili.
6	Najboljša in najučinkovitejša metoda posredovanja informacij razvojni ekipi in znotraj ekipe same, je pogovor iz oči v oči.
7	Delujoča programska oprema je primarno merilo napredka.
8	Agilni procesi promovirajo trajnostni razvoj. Sponzorji, razvijalci in uporabniki morajo biti zmožni konstantnega tempa za nedoločen čas.
9	Nenehna težnja k tehnični odličnosti in k dobremu načrtovanju izboljša agilnost.
10	Preprostost -- umetnost zmanjševanja količine nepotrebnega dela -- je bistvena.
11	Najboljše arhitekture, zahteve in načrti izhajajo iz tistih ekip, ki so samoorganizirane.
12	V rednih časovnih razdobjih ekipa išče načine, kako postati učinkovitejša ob rednem prilagajanju svojega delovanja.

Vir: [Bec01b]

Takoimenovani »Agilni dežnik«, ki zajema vse agilne metode, je predstavljen na Sliki 9 ([Hol06], [Lac10a], [Pro10a])[2].

Slika 9: »Agilni dežnik«



Vir: [Hol06], [Lac10a], [Pro10a]

Pred izrazom »agilne metode« je bil v uporabi izraz »lahke« metode (ang. lightweight), ki je nastal sredi 1990-tih, kot odgovor na »težke« metode (ang. heavywight), ki so značilne za takoimenovano »obdobje birokracije«. Teža metode je opredeljena kot produkt obsega in gostote metode, pri čemer je obseg metode odvisen od števila elementov, ki jih zajema, gostota pa je odvisna od ravni podrobnosti opisa elementov [Baj03]. «Težke« metode imajo velik obseg in veliko gostoto, saj za vsako aktivnost predpisujejo številne dokumente in obrazce. Agilne metode se pogosto opisujejo kot alternativa »planskim« ali »discipliniranim« metodam, kar daje napačen vtis, da so agilne metode »neplanske« ali »nedisciplinirane«. Nekateri celo izenačujejo agilne metode s »kavbojskim kodiranjem« (ang. »cowboy coding«) oziroma »kodiraj in popravi« programiranjem (ang. »code and fix«), za katerega je značilna odsotnost metod [Wik12a].

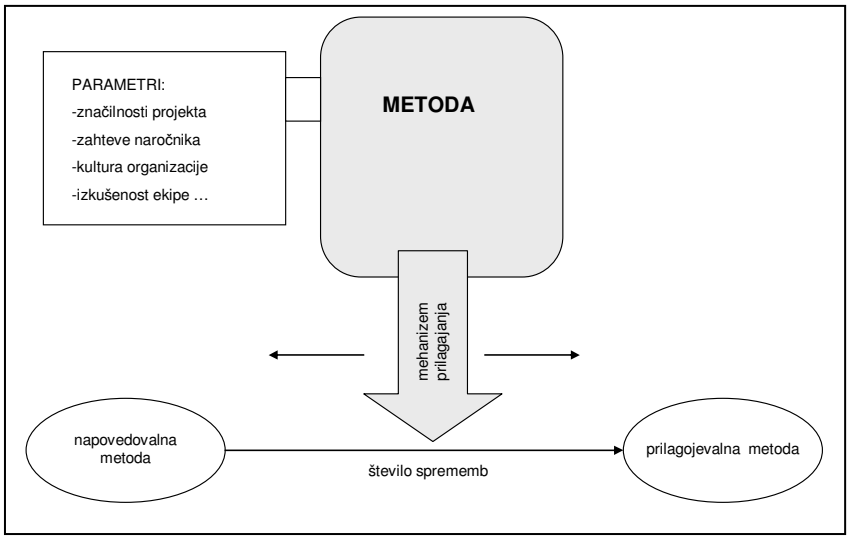
V Wikipediji kot bolj primerno razdelitev navajajo uvrščanje agilnih metod med »prilagojevalne« (ang. »adaptive«), tradicionalnih pa med »napovedovalne« (ang. »predictive«) metode, pri čemer je prilagodljivost neodvisna od teže metodologije. Za »prilagojevalne« metode je značilno, da planirajo podrobno le naslednjo iteracijo, za razliko od »napovedovalnih« metod, ki planirajo podrobno celotno izdajo programske opreme. Posledično so »prilagojevalne« metode bolj prijazne do sprememb, ki so pri »napovedovalnih« metodah časovno in stroškovno zahtevne, kar se odraža tudi pri poročanju poslovodstvu. Pri »napovedovalnih« metodah poslovodstvo lahko spremlja odmike od planov na nivoju celotne izdaje programske opreme, pri »prilagojevalnih« pa le na nivoju naslednje iteracije, vendar pa je pri »napovedovalnih« metodah rezultat viden pozneje kot pri

[2] V raziskavi Forrester's/Dr. Dobbs Global Developer Technographics Survey, Q3, 2009, v kateri je sodelovalo 1298 razvijalcev aplikativne programske opreme [Pro10a], je navedeno, da je najbolj razširjena agilno metodologija Scrum (11%), ki mu sledijo metode: Agilno modeliranje (»Agile Modeling«) (6%), FDD (»Feature Driven Development«) (3.8%), TDD (»Test-driven development«) (3.4%), ekstremno programiranje ali XP (2.9%), »Vitki« razvoj (»Lean development«) (2.1%), MSF (»Microsoft Solutions Framework for Agile) (1.8%), Agilne podatkovne metode (»Agile Data Method«) (1.6%), ASD (»Adaptive Software Development«) (1.3%), Six Sigma (0.9%), Crystal (0.3%), BDD (»Behavior-driven development«) (0.2%) in DSDM (»Dynamic Systems Development Method«) (0.2%). Wikipedija [Wiki10a] loči med agilnimi metodami in agilnimi praksami, ki jih uporablja več agilnih metod. Od zgoraj navedenih uvršča med agilne prakse naslednje: TDD in BDD.

»prilagojevalnih«, kjer so uporabni deli programske opreme na voljo po vsaki iteraciji, ki ponavadi traja od enega tedna do enega meseca [Wik12a].

Primernost uporabe »napovedovalne« ali »prilagojevalne« metode je odvisna od števila sprememb, kot je predstavljeno na Sliki 10.

Slika 10: Primernost uporabe »napovedovalne« ali »prilagojevalne« metode



Vir: [Baj03]

Za lažjo primerjavo med napovedovalnimi in prilagojevalnimi metodami lahko uporabimo tudi dve pojmovanji časa: »trdo«, v katerem se strogo držimo rokov in »mehko«, v katerem toleriramo odstopanja, vendar do meje, ki jo sami določimo. Npr. stereotipi germanskih držav poudarjajo pedantnost in urejenost, stereotipi romanskih pa ležernost in brezbrižnost, vendar so organizacijsko zahtevni projekti, kot npr. olimpijske igre, bili uspešno izvedeni tako v enih kot v drugih državah. Podobno lahko primerjamo fiksno in mobilno telefonijo: pri fiksni smo manj mobilni, vendar je kakovost signala boljša, pri mobilni pa imamo svobodo gibanja, vendar je kakovost signala odvisna od lokacije (npr. hribi) in okoliščin (npr. avtobus).

2.3.2 Primerjava agilnega in slapovnega razvoja programske opreme

Problemi na področju razvoja programske opreme so primerljivi s problemi na področju proizvodnje v letih 1960-1970 in sicer: prekoračitev rokov, prekoračitev stroškov, ustvarjanje funkcionalnosti, ki jih stranke ne želijo ter čedalje večji stroški na področju kontrole, ki pa ne prinašajo želenih učinkov [Str10].

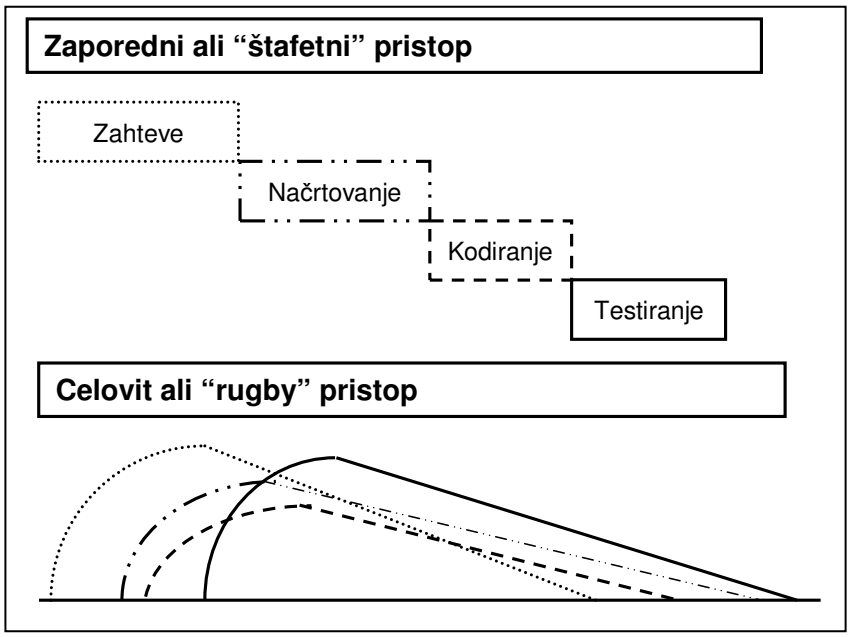
Čeprav je bil v industriji zaporedni oziroma slapovni pristop[3] k razvoju novih produktov opuščen že v 1990-tih, po vzoru japonskih proizvajalcev avtomobilov, to ni vplivalo na razvoj programske opreme, ki so ga IT managerji še naprej uvrščali med proizvodnjo po tekočem

[3] Slapovni model je objavil dr. Winston Royce leta 1970 v prispevku »Upravljanje razvoja velikih sistemov programske opreme«, pri čemer je navedel, da je opisani pristop zelo tvegan in je tudi priporočil vsaj dve Iteracijau, vendar pa je bil ta del prispevka pri uporabi te metode spregledan [Sza04]. Slapovni pristop nekateri avtorji [Sza04] primerjajo s slikanjem z zavezanimi očmi, ker uporabnik po opredelitvi zahtev ne sodeluje več pri razvoju in ne more sproti spremljati vmesnih rezultatov. Opisano situacijo pojasnjujejo s pravilom »KBVBV«, ki je akronim za pravilo »Ko bom videl, bom vedel« (oziroma »IKIWISI« kot akronim za »I'll Know It When I See It«), ki pravi, da uporabniki lahko bolje opredelijo zahteve potem, ko že vidijo in preizkusijo vsaj en del aplikacije.

traku in ne med razvoj novega produkta, ter posledično vztrajali na zaporednem ali slapovnem pristopu [Sza04].

Na Sliki 11 je predstavljena primerjava med slapovnim ali zaporednim pristopom, pri katerem si faze sledijo kot pri »štafeti« ter agilnim ali celovitim pristopom k razvoju novega produkta - programske opreme, pri katerem faze razvoja potekajo sočasno, kot pri »rugby«-ju, ko moštvo skupno premika žogo (»moving the scrum downfield«) ([Tak86], [Sza04]).

Slika 11: Zaporedni in celoviti pristop k razvoju novega produkta (programske opreme)



Vir: [Tak86], [Sza04]

Primerjava med tradicionalnim in agilnim pristopom pri projektih razvoja programske opreme je predstavljena v Tabeli 9.

Tabela 9: Tradicionalni in agilni pristop pri projektih razvoja programske opreme

Tradicionalni (Slapovni) pristop	**Agilni pristop**	**Praksa**
Časovnice	Končano je, ko je končano	Časovno okno – ne več kot X časa
Projektni vodje	Disciplinirane, samoorganizirane skupine	Sodelovanje med *Skrbnikom Metodologije Scrum* in naročnikom
Fiksni proračun	Upoštevanje sprememb pomeni, da stroški niso fiksni	Stroškovno okno – ne več kot X stroškov
Planirani izdelki, dostavljeni istočasno	Postopna dostava izdelkov, glede na njihovo vrednost in stalno učenje	Pred-projektni sestanki glede Uporabniških zgodb
Člani skupine se nahajajo na različnih lokacijah	Člani skupine se nahajajo na isti lokaciji	Člani skupine se sestanejo (fizično ali virtualno) ob ključnih dogodkih
Komunikacija z uporabo dokumentacije	Oglasne deske in razgovori	Elektronske oglasne deske in objavljeni dnevni sestanki
Naročnik je odstranjen	Naročnik je član skupine	Naročnik se sestabe s skupino ob ključnih dogodkih

Vir: po [PMI08a]

V nadaljevanju navajamo 5 mitov o agilnosti, ki se nanašajo na poenostavljeno pojmovanje agilnih metod ter pojasnila o dejanskem stanju [Hol06]:

- Mit 1: Agilni razvoj je nedisciplinirán.
  Uporaba agilnih praks, kot so »neprekinjeno integriranje« (»continuous integration«), testno voden razvoj (»test-driven development«), refaktoring (»refactoring«) in programiranje v parih (»pair-programing«), zahteva celo večjo discplino kot takoimenovane »disciplinirane« metode. Navedene agilne prakse so prisotne na način, ki je prilagojen razvijalcem in jih zato ne doživljajo kot administrativno breme oziroma oviro. Kot največji dokaz discipliniranosti pa avtor navaja kratka obdobja med izdajami delujoče, testirane programske opreme.
- Mit 2: Agilne skupine ne načrtujejo.
  Dejansko je obseg načrtovanja pri agilnih metodah celo večji kot pri takoimenovanih »načrtovalnih« metodah, vendar poteka celoten čas projekta – najprej na ravni izdaje, potem na ravni iteracije in končno na dnevni ravni (takoimenovano »just-in-time« planiranje [Sch11]), v primeru takoimenovanih »planskih« metod pa je planiranje v celoti izvedeno na začetku razvoja, ko je še zelo malo znanega.
- Mit 3: Agilni razvoj je nenapovedljiv.
  Podobno, kot pri mitu številka 2, tudi tukaj velja, da je obseg napovedovanja celo večji kot pri takoimenovanih »planskih« ali »disciplinirani« metodah, ker poteka celoten čas projekta, v primeru takoimenovanih »planskih« metod pa se napovedi izvajajo enkrat letno in so poznejše spremembe nezaželene. Pri agilnih metodah lahko razvijalci spremljajo pravilnost svojih ocen dnevno z uporabo Padajočega diagrama napredka (ang. »burndown chart«) in izračunom hitrosti skupine (ang. »velocity«), ob naslednji iteraciji pa lahko uporabijo zgodovinske podatke, kar vpliva na večjo zanesljivost napovedi.
- Mit 4: Agilni razvoj ni skalabilen (oziroma ni primeren za velike projekte).
  Stopnja tveganja pri velikih projektih je večja kot pri manjših, zato so agilne metode (pod določenimi pogoji) lahko celo bolj primerne za velike projekte, saj se napake in tveganja pokažejo hitreje, kot pri »tradicionalnih« metodah, ko je lahko že prepozno.
- Mit 5: Agilni razvoj je muha enodnevnica.
  Za muhe enodnevnice je značilno, da povzročajo veliko hrupa, vsebine pa je bolj malo. Raziskave ([Pro10a], [Ver11]) kažejo, da se obseg agilnega razvoja povečuje, predvsem pa se podjetja odločajo za »hibridne«pristope, v katerih prilagodijo agilni razvoj svojim posebnostim in potrebam.

Poenostavljene razlage agilnih pristopov so prispevale k zgoraj navedenim mitom, vendar se situacija izboljšuje, kar je razvidno tudi iz povečanja števila znanstvenih prispevkov na področju agilnih pristopov. V zadnjih desetih letih je dosežena kritična masa, ki daje agilnim pristopom kredibilnost in ne podpihuje lažnih sanj o »srebrni krogli« (ang. »silver bullet«), ki bo magična rešitev za vse probleme, temveč jih uvršča v množico orodij, s katerimi lahko zmanjšujemo tveganja pri razvoju programske opreme. Večji obseg izkušenj na področju agilnih pristopov prispeva k doseganju višjih stopenj zrelosti in posredno tudi k doseganju višjih stopenj uspešnosti.

2.3.3 Uspešnost agilnega razvoja programske opreme

Rezultati raziskave Standish Group „Chaos Manifesto 2011“ [Coh12] so pokazali, da je odstotek uspešnih agilnih projektov (42%) trikrat večji od odstotka uspešnih slapovnih projektov (14%). Podobno v svoji raziskavi za leto 2011 ugotavlja [Amb11], pri čemer je odstotek uspešnih slapovnih projektov enak kot pri ad-hoc projektih (40%), odstotek iterativnih, agilnih in vitkih projektov pa je večji (60%).

Stopnje uspešnosti so stabilno naraščale od 1994 do 2000, ko so se znižale, potem pa so zopet začele rasti v letih od 2002 do 2006. V letu 2004 je bila neuspešnih samo 15 % projektov [Sza04], kar je bistveno izboljšanje v primerjavi z 31 % v letu 1994. Jim Johnson, predsednik uprave Standish Group, je podal mnenje, da je izboljšanje posledica manjših projektov, ki uporabljajo iterativne metode [Sza04]. Povečanje stopnje neuspešnosti na 24% v letu 2010 je Jim Johnson, pojasnil kot posledico gospodarske krize [Lev09], v kateri podjetja hitreje in lažje stornirajo projekte, ki niso nujni oziroma niso uspešni. Leta 2010 je bila v reviji IEEE Software objavljena kritika metode, ki je uporabljena pri Chaos poročilih [Gat10], pri čemer so izpostavljeni štirje problemi: (1) omejenost na stroške, čas in funkcionalnost; (2) nerealne stopnje uspešnosti, ki so posledica enostranskosti merila za točnost napovedi; (3) neustreznost definicij; (4) primerjava številk, ki temeljijo na različnih predpostavkah. Avtorji članka so uporabili definicije skupine Standish na lastni bazi podatkov (5.457 napovedi za 1.211 projektov), njihovi rezultati pa so se razlikovali od rezultatov skupine Standish.

Agilni pristop k razvoju programske opreme uporablja 35% anketiranih podjetij, kot je navedeno v raziskavi Forrester’s/Dr. Dobbs v letu 2009, v kateri je sodelovalo 1298 razvijalcev aplikativne programske opreme [Pro10a]. V isti raziskavi je navedeno, da 21% razvijalcev uporablja iterativne metode, 13% slapovne metode, preostalih 31% pa ne uporablja formalnih metod. Rezultati navedene raziskave so pokazali, da 27% podjetij uporablja samo eno metodo, 63% pa uporablja mešanico različnih metod ali mešanico agilnih in neagilnih metod. Gartnerjeva analitika (Thomas Murphy in David Norton) sta predvidevala [Pro10b], da bodo do leta 2012 agilne metode uporabljene pri 80% vseh projektov razvoja programske opreme.

V strategiji za informacijsko komunikacijsko tehnologijo britanske vlade, objavljene leta 2011, je navedeno, da bo uporabljala agilne metode pri nabavi in razvoju informacijsko komunikacijske tehnologije (IKT) z namenom znižanja tveganja neuspešnosti projektov [NAO12]. Kot primeri uporabe agilnih metodologij so navedena naslednja podjetja: British Airways plc, BT plc, Innocent Drinks, IBM United Kingdom Limited, Vodafone Group plc. Inštitut za Vlado (ang. Institute for Government) je priporočil vladi, da izvaja veliko agilnih pilotskih projektov, z namenom pridobivanja izkušenj in izmenjave znanja. Prvi agilni projekt je bil prototip za alpha.gov.uk spletno stran. Strateški cilji britanske vlade vključujejo [HMG11] uporabo agilnega pristopa v polovici IKT programov za spremembe do aprila 2013 ter zmanjšanje povprečnega časa dobave IKT programov za spremembe za 20% do 2014.

2.4 Metodologija *Scrum*

2.4.1 Uvod

Izraz *Scrum*[4] je bil prvič uporabljen pri opisu razvojnega procesa leta 1986, ko sta Hirotaka Takeuchi and Ikujiro Nonaka objavila članek v reviji Harvard Business Review [Tak86], v katerem sta opisala celovit pristop k razvoju novih produktov kot rugby moštvo, pri katerem se vsi člani moštva premikajo proti cilju skupaj v različnih smereh (ang. »moving the scrum downfield«), za razliko od zaporednega ali »štafetnega« pristopa, pri katerem en član moštva podaja štafeto naslednjemu članu moštva.

Izraz *Scrum* sta v kontekstu razvoja programske opreme leta 1991 prvič uporabila DeGrace in Stahl [DeG90]. Leta 1993 sta *Scrum* uporabila pri razvoju programske opreme istočasno, vendar neodvisno, Ken Schwaber v svojem podjetju Advanced Development Method (ADM) ter Jeff Sutherland, John Scumniotles in Jeff McKenna v podjetju Easel [Sut04]. Leta 1995 sta Sutherland in Schwaber skupaj predstavila prispevek z opisom metodologije *Scrum* na konferenci OOPSLA '95 v Austinu, Texas, leta 2002 Sta Ken Schwaber in Mike Beedle izdala knjigo »Agilni razvoj programske opreme z uporabo metodologije *Scrum*« [Sch02], leta 2004 pa je Ken Schwaber izdal knjigo »Agilno vodenje projektov z uporabo metodologije *Scrum*« [Sch04].

Skupne značilnosti športa rugby in metodologije *Scrum* so [Dru10], da je moštvo/skupina pomembnejše od posameznika, da so moštva/skupine večdiscpilinarna, da se hitro prilagajajo in da je skupinska kultura visoko razvita[5].

Avtorja metodologije *Scrum* navajata ([Sut09], [Sch11]), da *Scrum* ni metoda, temveč ogrodje (ang. »framework«)[6] za razvoj kompleksnih izdelkov. Zaradi lažjega razumevanja bomo v nadaljevanju uporabljali pojem metodologija *Scrum.*

Metodologija *Scrum* temelji na empirični procesni kontrolni teoriji in vključuje iterativen in inkrementalen proces za optimiziranje napovedovanja in kontrolnega tveganja [Sch11]. Za implementacijo empirične procesne kontrole so značilni trije stebri:

- Preglednost (»Transparency«) – elementi procesa, ki vplivajo na izhod morajo biti pregledni udeležencem, ki upravljajo izhod;
- Preverjanje (»Inspection«) – pogostost preverjanja mora zagotavljati pravočasno odkritje nesprejemljivih odstopanj, pri tem je potrebno upoštevati toleranco do preverjanja;
- Prilagajanje (»Adaptation«) – v primeru nesprejemljivih odstopanj je proces ali izdelek pravočasno prilagoditi.

---

[4] V angleško-slovenskem slovarju najdemo naslednji opis: »scrum« [skram] je krajšava za »scrummage«, kar je inačica od »scrimmage«, ki v ameriški anlgeščini pomeni prerivanje za žogo. V Wikipediji je navedeno, da je Scrum izraz, ki v rugby-ju pomeni vrnitev igre v žogo, ko se šest ali osem članov vsakega tima postavi v tri vrste s stisnjenimi glavami, tako, da tvorijo »tunel«, znotraj katerega se potem borijo za žogo tako, da jo z nogami premikajo nazaj, pri čemer poskušajo premakniti nasprotni tim čim bolj nazaj. Opazovalci vidijo gručo igralcev, ki se premika po igrišču skupaj.

[5] Avtorica [Dru10] navaja znani rek, da je ameriški rugby sport gentlemanov, ki ga igrajo huligani, angleški rugby pa je sport huliganov, ki ga igrajo gentlemani. Za igralce rugby-ja velja strog etični kodeks na igrišču in izven igrišča, za katerega so značilni: »timsko delo, spoštovanje, dobra volja in fair-play«, podobno vrednotam, ki jih promovira Scrum: »predanost, usmerjenost, odprtost, spoštovanje in pogum«.

[6] V tem delu smo angleški izraz »framework« prevedli kot ogrodje, ki je v slovenskem knjižnem slovarju opredeljeno kot med seboj povezani bistveni sestavni elementi česa. Schwaber [Sch10c] je v intervjuju izjavil, da Scrum ni metodologija, ker ne predpisuje rešitev za vse možne primerem temveč temelji predvsem na posameznikih, ki poiščejo ustrezno rešitev glede na okoliščine.

Navedeno je v veliki meri primerljivo z Demmingovim ciklom Načrtuj-Izvedi-Preveri-Ukrepaj (»Plan-Do-Check-Act«).

Metodologija *Scrum* je opisana v knjigi, izdani 2002 [Sch02], dopolnitve pa so objavljene v takoimenovani knjigi znanja (»book of knowledge«), ki jo vzdržujeta avtorja metodologije *Scrum* [Sch11].

Rezultati raziskave podjetja Version One [Ver11], v kateri je sodelovalo 6.042, ki kažejo, da sta dve najbolj razširjeni agilni metodi metodologija *Scrum,* ki jo uporablja 50% anketirancev in *Scrum/XP Hibrid,* ki jo uporablja 14% anketirancev.

2.4.2 Opis metodologije *Scrum*

V nadaljevanju je podan opis postopka razvoja programske opreme z uporabo metodologije *Scrum*[7], ki temelji na obeh virih in je prikazan na Sliki 12. Značilna za Sliko 12 je dvojna zanka, ki je sicer simbol metodologije *Scrum*, pri čemer velja naslednje:

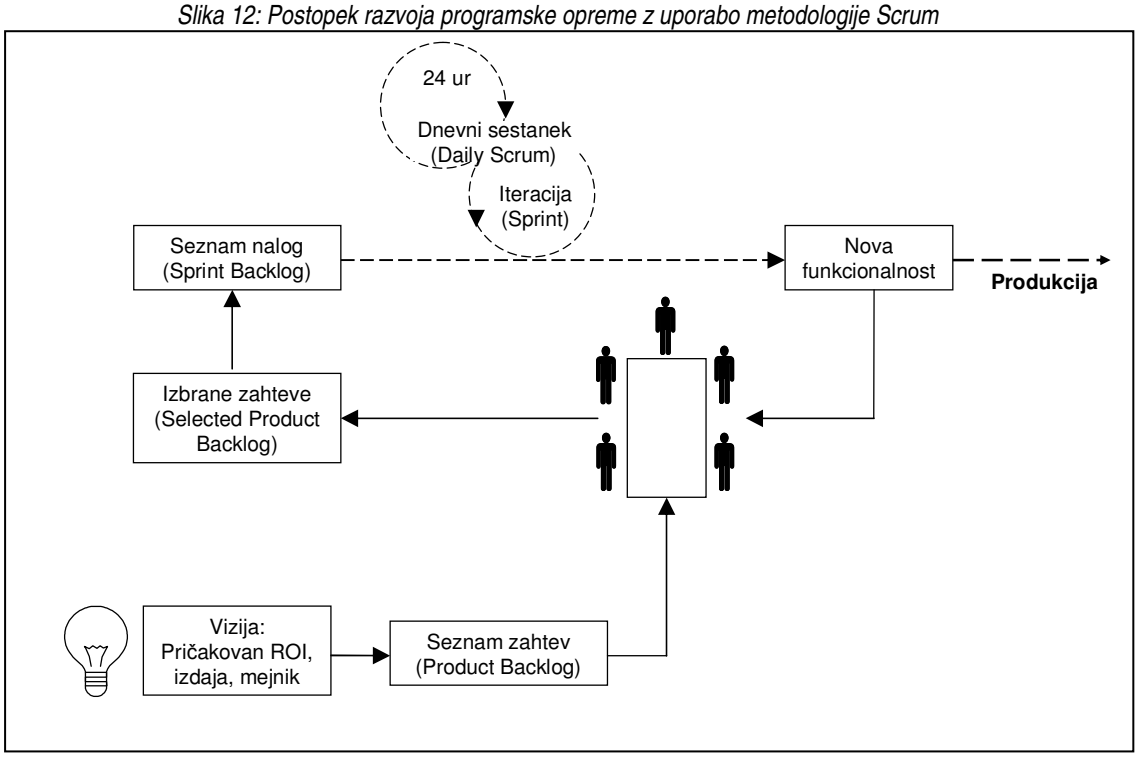
- prva zanka pomeni manjši cikel ali *Scrum* (npr. en dan, oziroma obdobje med *Dnevnima Scrum sestankoma*,
- druga zanka pa pomeni daljši cikel ali *Iteracijo* (npr. en mesec, oziroma obdobje od *Načrtovalnega sestanka Iteracije* do *Potrjevalnega sestanka Iteracije*).

Pri opisu metodologije *Scrum* se pogosto uporablja tudi izraz »time-box« ali časovno okno, ki se nanaša na dolžino zanke ali *Iteracije*, ki predstavlja ritem skupine ([May09b] oziroma srčni utrip skupine.

Razvoj poteka v več *Iteracijah*; posamezna *Iteracija* naj bi trajala predvidoma od enega tedna do največ 30 koledarskih dni. Na začetku procesa *Produktni vodja*, ki zastopa interese uporabnikov in skrbi za financiranje projekta, pripravi *Seznam Uporabniških zgodb* in določi prioriteto *Zahtev.* V zadnjem objavljenem opisu metodologije *Scrum* [Sch11] je navedeno, da je *Seznam Uporabniških zgodb* (ang. »Product Backlog«) sestavljen iz *Uporabniških zgodb*. *Produktni vodja* je odgovoren za poslovno vrednost *Seznama Uporabniških zgodb*. Vlogo *Produktnega vodje* bi lahko poimenovali tudi »vloga *Kaj*« [May09b], ker odloča o tem, *Kaj* se bo naredilo. *Seznam Uporabniških zgodb* je dinamičen in se prilagaja potrebam naročnika tako, da *Produktni vodja* dodaja nove *Uporabniške zgodbe* ter jih razvršča glede na poslovne prioritete, pri čemer velja, da *Seznama Uporabniških zgodb* ni dovoljeno spreminjati v času *Iteracije* oziroma je v času *Iteracije* »zamrznjen« (takoimenovani »time-box« ali časovno okno, v katerem ni sprememb).

[7] Literatura iz področja agilnih metodologij je večinoma v angleščini. Večji del literature je dostopen na spletu in je prav tako v angleščini. V slovenščini nismo zasledili literature, ki bi jo lahko uporabili kot referenco pri prevodu pojmov, ki jih uporablja metodologija Scrum. Osnovne Scrum pojme smo poiskali tudi v islovarju (www.islovar.org), vendar prevodi zaenkrat še niso dostopni. Zato smo se odločili, da vpeljemo naš prevod pojmov, ki smo jih navedli v Pojmovniku. Z namenom, da bi povečali razumljivost disertacije, pa smo uporabljene *Scrum* pojme zapisali v *poševni pisavi*.

Slika 12: Postopek razvoja programske opreme z uporabo metodologije Scrum



Vir: [Schw04]

Produktni vodja in člani *Razvojne skupine* se na začetku vsake *Iteracije* sestanejo na *Načrtovalnem sestanku Iteracije,* na katerem izberejo podmnožico *Uporabniških zgodb*, ki bo realizirana v tej *Iteraciji. Načrtovalni sestanek Iteracije* traja osem ur in je sestavljen iz dveh delov, pri čemer posamezni del traja štiri ure. Dolžina *Načrtovalnega sestanka Iteracije* je odvisna od dolžine *Iteracije.* Za krajše *Iteracije* so predvideni krajši *Načrtovalni sestanki Iteracije*. V splošnem velja, da dolžina *Načrtovalnega sestanka Iteracije* ne sme presegati 10% dolžine *Iteracije* [Sut09a]). V prvem delu *Načrtovalnega sestanka Iteracije*, poimenovanem tudi »Del *Kaj*«, se *Produktni vodja* in *Razvojna skupina* dogovorita, katere *Uporabniške zgodbe* iz *Seznama Uporabniških zgodb* bodo vključene v tekoči *Iteraciji*. V drugem delu *Načrtovalnega sestanka Iteracije,* poimenovanem tudi »Del *Kako*«, *Razvojna skupina* izdela *Seznam Nalog Iteracije,* ki vsebuje *Naloge*, potrebne za realizacijo dogovorjenih *Uporabniških zgodb,* pri čemer je časovna zahtevnost *Nalog* omejena na štiri do 16 ur dela. Pri ocenjevanju zahtevnosti realizacije *Uporabniških zgodb* lahko *Razvojna skupina* uporablja metodo *Načrtovalni poker* [Gre02] *Seznam Nalog Iteracije* je, podobno kot *Seznam Uporabniških zgodb*, dinamičen in se dopolnjuje po *Dnevnem Scrum sestanku*. *Razvojno skupino* so nekateri avtorji poimenovali tudi »*Pleme Kako*« [May09b], ker odloča o tem, *kako* se bo naredilo tisto, kar je izbral *Produktni vodja*, poimenovan »*Glas Kaj*«.
Na *Dnevnih Scrum sestankih*, ki trajajo največ 15 minut, posamezni član *Razvojne skupine* odgovori na naslednja tri vprašanja:

- Kaj sem naredil(a) v času od prejšnjega *Dnevnega Scrum sestanka*?
- Kaj bom naredil(a) do naslednjega *Dnevnega Scrum sestanka*?
- S katerimi *Ovirami* se srečujem pri delu?

Na opisani način je zagotovljena transparentost in možnost za usklajevanje znotraj *Razvojne skupine*. Naloga *Skrbnika Metodologije Scrum* je predvsem, da *Razvojni skupini* pomaga doseči cilj na način, da:

- zagotavlja, da *Razvojna skupina* razvija samo tiste *Uporabniške zgodbe*, ki so v *Seznamu Uporabniških zgodb*,
- rešuje *Ovire,* ki jih člani *Razvojne skupine* izpostavijo na *Dnevnih Scrum sestankih*,
- zagotavlja, da *Razvojna skupina* upošteva pravila metodologije *Scrum.*

Ob koncu *Iteracije* se izvede *Potrjevalni sestanek Iteracije*, na katerem *Razvojna skupina* predstavi *Produktnemu vodji* in vsem zainteresiranim uporabnikom rezultate tekoče *Iteracije*. *Produktni vodja* na *Potrjevalnem sestanku Iteracije* preveri ustreznost izdelka in posreduje svoje predloge in pripombe. Dolžina *Potrjevalnega sestanka Iteracije* je omejena na največ štiri ure.

Po *Potrjevalnem sestanku tekoče Iteracije* in pred *Načrtovalnim sestankom naslednje Iteracije*, se izvede *Retrospektivni sestanek Iteracije*, na katerem *Razvojna skupina* in *Skrbnik Metodologije Scrum* kritično ocenijo razvojni proces v tekoči *Iteraciji* in se dogovorijo za izboljšave, ki jih bodo upoštevali v naslednji *Iteraciji*. Pri tem posamezni član *Razvojne skupine* odgovori na naslednja vprašanja [Schw02]:

- Kaj je bilo pri tej *Iteraciji* narejeno dobro?
- Kaj bi bilo potrebno izboljšati pri naslednji *Iteraciji*?

Dolžina *Retrospektivnega sestanka Iteracije* je omejena na največ tri ure.

Poenostavljen opis metodologije *Scrum*, ki bi ga lahko poimenovali tudi »3-6-4« je predstavljen na Sliki 13 ([Sch11], [Scr12], [Mou10]):

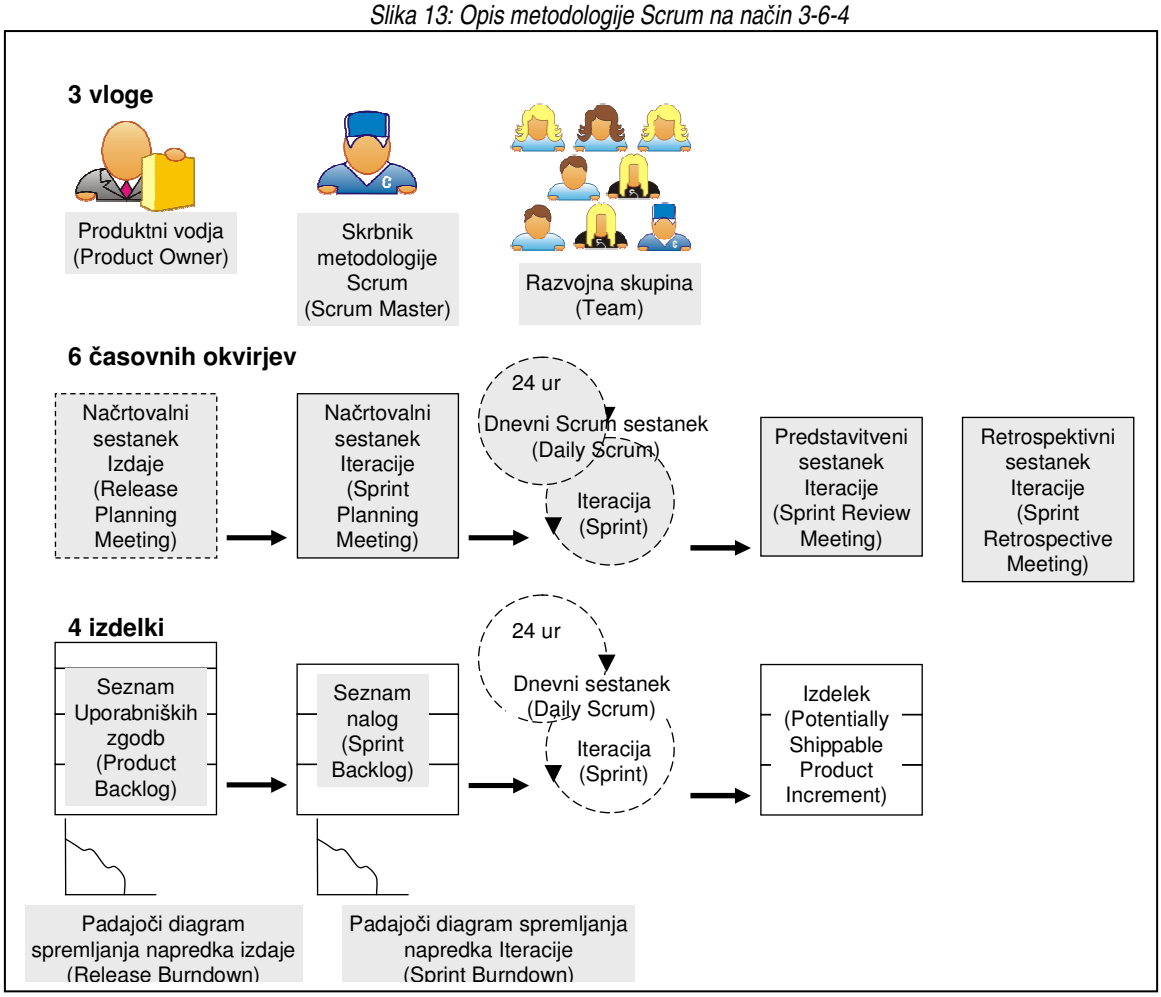
- 3 vloge (*Produktni vodja*, *Skrbnik Metodologije Scrum, Razvojna skupina*);
- 6 časovnih oken (*Načrtovalni sestanek Izdaje, Iteracija, Načrtovalni sestanek Iteracije, Dnevni Scrum sestanek, Potrjevalni sestanek Iteracije, Retrospektivni sestanek Iteracije*);
- 4 izdelki (*Seznam Uporabniških zgodb in Padajoči diagram spremljanja napredka Izdaje, Seznam Nalog Iteracije in Padajoči diagram spremljanja napredka Iteracije).*

Avtorja metodologije *Scrum* [Sch11] navajata, da je *Načrtovalni sestanek Izdaje* opcijski, nanaša pa se na določanje ciljev *Izdaje* in najvišjih prioritet postavk *Seznama Uporabniških zgodb Izdaje*. *Padajoči diagram spremljanja napredka Izdaje* prikazuje oceno preostalega napora na nivoju *Izdaje*, pri čemer *Razvojna skupina* določi enoto mere. *Padajoči diagram spremljanja napredka Iteracije* prikazuje oceno preostalega napora na nivoju *Dnevnih Scrum sestankov*, pri čemer *Razvojna skupina* določi enoto mere. Oba diagrama spremljanja napredka sta ves čas razvoja vidna vsem udeležencem projekta.

Tudi za Sliko 13 je značilna dvojna zanka, ki poudarja iterativnost in inkrementalnost metodologije *Scrum*. Poenostavljeno bi lahko metodologijo *Scrum* opisali na naslednji način: Na *Načrtovalnem sestanku Iteracije Produktni vodja* opredeli trenutne *Zahteve* ali *Uporabniške zgodbe* in jim določi prioriteto, kar zapiše v *Seznam Uporabniških zgodb*, potem *Skrbnik Metodologije Scrum* in *Razvojna skupina* določijo *Seznam Nalog Iteracije,* ki vsebuje *Naloge* za tiste *Zahteve* ali *Uporabniške zgodbe* iz *Seznama Uporabniških zgodb*, ki jih želi *Produktni vodja* imeti ob zaključku tekoče *Iteracije*. *Razvojna skupina* in *Skrbnik Metodologije Scrum* na *Dnevnem Scrum sestanku* ugotavljajo, kaj je bilo narejeno v času od predhodnega *Dnevnega Scrum sestanka*, kaj bo narejeno v času do naslednjega *Dnevnega Scrum sestanka* ter morebitne *Ovire*, v *Padajočem diagramu napredka Iteracije* pa narišejo oceno števila preostalih ur do zaključka *Iteracije*. Po zaključku *Iteracije Razvojna skupina* na *Potrjevalnem sestanku Iteracije* predstavi izdelek *Produktnemu vodji*, ki poda povratno

informacijo o ustreznosti izdelka. Če je izdelek ustrezen, ga *Produktni vodja* potrdi. Potem sledi *Retrospektivni sestanek Iteracije*, na katerem *Skrbnik Metodologije Scrum* in *Razvojna skupina* ugotavljajo prednosti in pomanjkljivosti predhodne *Iteracije*, ter pridobljene izkušnje upoštevajo v naslednji *Iteraciji.*

Slika 13: Opis metodologije Scrum na način 3-6-4

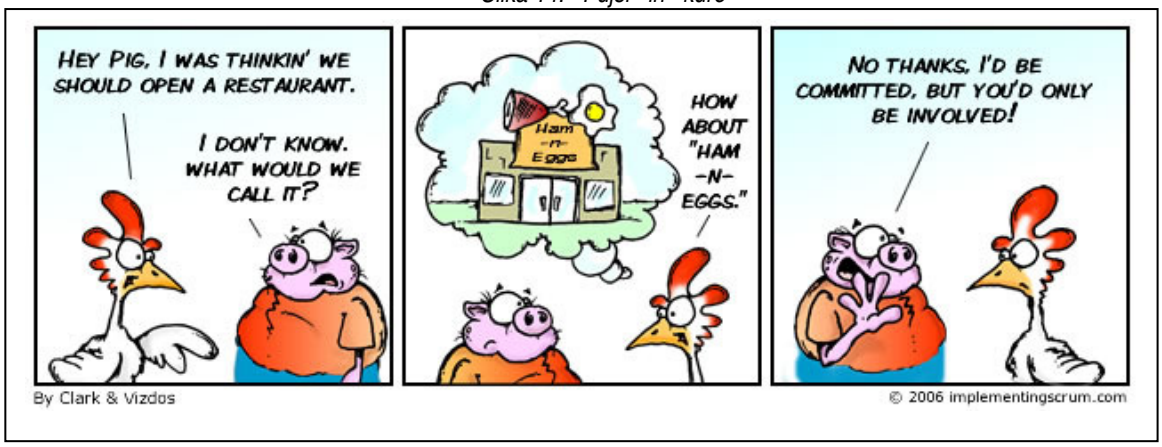


Vir: po [Sch11], [Scr12], [Mou10]

2.4.3 Vloge pri metodologiji Scrum

Za razumevanje metodologije Scrum je pomembno pojasniti delitev vlog na »pujse« (ang. »pigs«) in »kure« (ang. »chickens«), pri čemer so vloge poimenovane po šali, prikazani na sliki 14.

Slika 14: »Pujsi« in »kure«



Razumevanje šale v izvirniku je enostavno, v slovenščini pa je potrebno dodatno pojasnilo. Vloga »pujsa« se nanaša na tiste vloge, ki so popolnoma predane nalogi. Analogija s »pujsom« pomeni, da za šunko z jajci (ang. »ham and eggs«) mora »pujs« žrtvovati svoje življenje oziroma, da vloga »pujsa« tvega izgubo službe ob neuspehu projekta. *Produktni vodja* ima vlogo »pujsa« za *Seznam Uporabniških zgodb*, *Skrbnik Metodologije Scrum* ima vlogo »pujsa« za *Scrum* projekt, *Razvojna skupina* ima vlogo »pujsa« za razvito programsko opremo. Vloga »kure« se nanaša na vse ostale udeležence, ki sodelujejo pri projektu, vendar ne tvegajo izgube službe ob neuspehu projekta. Analogija s »kuro« pomeni, da za šunko z jajci »kura« prispeva le jajce in ji ni potrebno žrtvovati življenja oziroma, da vloga »kure« ne tvega izgube službe ob neuspehu projekta (npr. uporabniki). Pri metodologiji Scrum velja pravilo, da »kure« ne morejo vplivati na način dela »pujsov«. »Kure« so lahko prisotne na *Dnevnih Scrum sestankih*, vendar lahko le poslušajo in nimajo pravice posegati v delo *Razvojne skupine*. V nadaljevanju naloge bomo uporabljali pojma »pujs« in »kura« na predhodno opisani način.

Povzetek opisa *Scrum* vlog glede ne delitev med »pujse« in »kure« je podan v Tabeli 10.

Tabela 10: Scrum vloge

Vloga	Opis	Komentar
Produktni vodja	- Predstavnik naročnika - Ustvari in vzdržuje *Seznam Uporabniških zgodb* - Odloča v imenu vseh naročnikov in uporabnikov - Predstavi in pojasni *Seznam Uporabniških zgodb Razvojni skupini*	»Pujs« za *Seznam Uporabniških zgodb*
Skrbnik Metodologije Scrum	- Vodi agilno izvrševanje naloge - Odgovoren za razvojni proces - Odgovoren za maksimiziranje produktivnosti *Razvojne skupine* - Pripravlja in vodi sestanke - Predstavnik za vodstvo in *Razvojna skupina* - Lastnosti trenerja	»Pujs« za razvojni proces
Razvojna skupina	- Razvija zahteve, ki jih določi naročnik - Samoorganizirano - Sedem plus minus dva člana - Poslovne in tehnične sposobnosti za razvoj funkcionalnosti - Odgovorno za ocenjevanje in predano delu - Popolna avtonomija in avtoriteta v času *Iteracije*	»Pujs« za razvito programsko opremo
Ostali udeleženci projekta		»Kure«

Vir: po [PMI08a]

3. MERJENJE UČINKOVITOSTI AGILNEGA RAZVOJA PROGRAMSKE OPREME

3.1 Raziskave agilnih metod

Agilni razvoj programske opreme je predmet raziskav več kot 10 let, od objave agilnega manifesta leta 2001. V nadaljevanju bomo podali pregled ključnih raziskav in smernic za nadaljnje raziskave, ki so pomembni za razvoj modela spremljanja učinkovitosti agilnega razvoja programske opreme z uporabo metodologije *Scrum*.

Dolgo je veljalo, da je število raziskav na področju agilnega razvoja programske opreme skromno. Prvi temeljitejši pregled agilnih metod je podal Abrahamson [Abr02], ki je podal kriterij za ugotavljanje agilnosti metode, opis najbolj pogostih agilnih metod ter primerjavo metod glede na pokritost faz razvoja programske opreme, velikost tima in druge lastnosti. Pri tem ugotavlja, da večina trditev o uspešnosti agilnih metod temelji na zgodbah o uspehu in ne na empiričnih raziskavah.

Iz sistematičnega pregleda empiričnih raziskav na tem področju pred letom 2005 [Dyb08] je razvidno, da obstaja pomanjkanje teoretičnega in metodološkega pristopa. Avtorja sta ugotovila, da je bilo do leta 2005 objavljeno 33 primarnih empiričnih raziskav, od tega jih je 13 (39%) kot raziskovalno metodo uporabilo študijo enega primera. Pri tem je bilo devet primerov iz gospodarstva in štirje primeri iz akademskega okolja, od katerih so trije uporabili isti projekt. V samo enem primeru so udeleženci že imeli izkušnje z uporabo agilnih metod. Študija več primerov je bila uporabljena v 11 raziskavah (33%), pri čemer so vsi primeri bili iz gospodarstva, vendar so bili udeleženci brez predhodnih izkušenj. Od skupaj 33 empiričnih raziskav, je bila večina izvedena za metodo XP (25 študij ali 76%) in le ena za metodologija *Scrum* (3%). Avtorja sta predlagala, da bi raziskovalci več pozornosti namenili metodologiji *Scrum*.

Dingsøyr et al. ([Din08], menijo, da obstaja velik razkorak med razširjeno uporabo agilnih metodologij v praksi in majhnim številom empiričnih raziskav, ter predlagajo teme, ki naj bi jih raziskovalci prioritetno obravnavali naslednjih pet do deset let. Zrelost tedanjega stanja na področju raziskav metodologije *Scrum* so ocenili z najnižjo stopnjo zrelosti: »v nastajanju« (ang. nascent). Kot predmet raziskav do leta 2015 predlagajo metodologijo *Scrum* ter vpliv agilnih metod na projektno vodenje.

Pregled raziskav v obdobju 2001-2010 je podan v [Din12], kjer so avtorji izvedli pregled literature, objavljene v »ISI Web of Science« na področju agilnega razvoja programske opreme med vključno letoma 2001 in 2010, glede na države ter revije in konference. Rezultati pregleda so pokazali, da se je število objav povečalo in da večina raziskav temelji na praktičnih izkušnjah. V letih 2008 in 2009 je bilo letno objavljeno več kot 275 prispevkov. Avtorji prispevkov v obsegu raziskave so bili iz 63 različnih držav, pri čemer so največ objav imele Združene države Amerike (338 objav ali 21,7%), Kanada (110 objav ali 7,1%), Nemčija (96 objav ali 6,2%), Finska (94 objave ali 6,0%) in Združeno kraljestvo (94 objav ali 5,3%). Slovenija se je s petimi objavami uvrstila na 36. mesto (0,3%). Raziskava je pokazala, da so najbolj popularne konference na tem področju »International Conference on Agile Software Development (XP)« v Evropi in »Agile« v Združenih državah Amerike. Največ prispevkov je

bilo objavljenih v naslednjih revijah: »IEEE Software«, »Journal of Systems and Software«, »Information and Software Technology« in »Empirical Software Engineering«. Avtorji so ugotovili, da je v zadnjem času največ pozornosti namenjeno projektnemu vodenju agilnih projektov. Objavljeno je večje število rešitev, vendar manjkajo empirične raziskave, ki bi potrdile njihov veljavnost. Avtorji so ugotovili, da je akademska skupnost na začetku raziskovala predvsem ekstremno programiranje, potem pa je preusmerila pozornost na metodologija *Scrum*.

Primerjava med podjetji, ki za razvoj programske opreme uporabljajo agilni pristop, in podjetji, ki se poslužujejo tradicionalnega discipliniranega pristopa je pokazala, da uporaba agilnega pristopa izboljša vodenje projektov in odnose z uporabniki [Ces05], zmanjšuje obseg nadurnega dela in prispeva k večjemu zadovoljstvu uporabnikov [Man05].

Zaradi pomanjkanja raziskav v gospodarskem okolju smo upoštevali tudi raziskave v akademskem okolju. Uporaba projektnega dela pri poučevanju predmetov, ki obravnavajo agilni razvoj programske opreme, je pogosta. Nekateri primeri so: poučevanje ekstremnega programiranja [Shu02], preverjanje učinkovitosti testno vodenega razvoja in programiranja v parih [Xu06] ter poučevanje razlik med agilnim in discipliniranim pristopom k razvoju programske opreme [Rob08]. Pravilno načrtovana študija [Car10] omogoča, da se ugotovijo prednosti in pomanjkljivosti modela v akademskem okolju in zmanjšajo tveganja pri uporabi modela v gospodarskem okolju. Za sodelovanje študentov ob zaključku študija se pričakuje, da je primerljivo s sodelovanjem strokovnjakov začetnikov, kar je opisano v študiji [Run03]. Študenti so pozitivno ocenili predmet, ki jim je omogočil, da so se naučili agilnih metod z uporabo projektnega pristopa, ki se je pokazal kot uporaben tudi na drugih inženirskih področjih [Val10]. Predhodne študije odziva študentov na uporabo agilnih metod [Mel05] so pokazale, da se študenti ne počutijo suverene na področju planiranja in ne zaupajo svojim napovedim. Veliko študentov meni, da je načrtovanje in napovedovanje neproduktivno administrativno delo ter se ne zaveda pomena ustreznih načrtov in napovedi.

3.2 Zrelost agilnih metod

3.2.1 *Scrum* zrelostni modeli

Primerjava med metodologijo *Scrum* in CMMI je podana v ([Chi04]. Poglobljena primerjava metodologije *Scrum* in CMMI za področje projektnega vodenja je podana v [Pre12]. Rezultati primerjave kažejo, da *Scrum* projektni procesi in prakse na področju projektnega vodenja izpolnjujejo v celoti zahteve CMMI na ravni dve in delno na ravni tri. Raven štiri je možno doseči z uvedbo agilnih metrik, za prehod na raven pet pa je potrebno zmanjšati zahtevnost in povečati organizacijsko in procesno učinkovitost.

Model za presojo agilnih metod (predvsem XP) po standardu ISO 9001 je podana v [Qas12].

Organizacije se razlikujejo glede zrelosti uporabe agilnih metod, pri ocenjevanju zrelostne stopnje pa je možno uporabiti različne modele ([Pro10c], [Amb10]). V Wikipediji [Wik12a] je navedenih nekaj možnosti za merjenje zrelosti agilnosti, npr. Agility Index Measurement (AIM), pri katerem se sešteva vrsto agilnih faktorjev, Agility Measurement Index, ki upošteva pet dimenzij projektov programske opreme (trajanje, tveganje, obseg novosti, napor,

medsebojno sodelovanje), agilne samoocenitve (npr. Nokia test, Karlskrona test, 42 točk test ipd). Navedene možnosti so zaenkrat neformalnega značaja. V literaturi najdemo tudi naslednje možnosti: Sidky Agile Measurement Index (SAMI) [Sid07], Agile Adoption and Improvement Model (AAIM).

Martin Proulx [Pro10c] je v svojem modelu AMM (ang. Agility Maturity Model) povezal rezultate in značilnosti posameznih zrelostnih stopenj, pri čemer je imel za vodilo načelo, da višja stopnja zrelosti pomeni boljše rezultate pri zadovoljstvu uporabnikov, učinkovitosti in stroških. Njegov model temelji na metodologiji *Scrum*, kot najbolj razširjenem agilnem ogrodju, povzetek ključnih značilnosti posameznih stopenj pa je podan v Tabeli 11:

- Prva ali skupinska stopnja (ang. Team level), v kateri skupina uporablja metodologijo *Scrum*, ne da bi pridobilo dovoljenje s strani nadrejenih, pri čemer uporablja le nekatere prakse, večinoma nedosledno;
- Druga ali oddelčna stopnja (ang. Department level), v kateri več skupin uporablja metodologijo *Scrum*, vodstvo pa je opazilo izboljšanje produktivnosti skupin, ki uporabljajo metodologijo *Scrum*;
- Tretja ali poslovna stopnja (ang. Business level), ko se poslovna stran vključuje v delo *Razvojne skupine* in se sodelovanje s poslovno stranjo povečuje;
- Četrta ali projektna stopnja (ang. Project management level), ko je metodologija *Scrum* integrirano v projektno vodenje, ki pa še vedno večinoma sloni na priporočilih PMBOK (Project Management Book of Knowledge);
- Peta ali poslovodna stopnja (ang. Management level), v kateri je poslovodstvo sprejelo agilne pristope ter ustrezno prilagodilo organizacijo ter proces poročanja in
- Šesta ali organizacijska stopnja (ang. Corporate-wide level), ko je agilnost razširjena v celotnem podjetju (avtor jo poimenuje tudi »Nirvana« stopnja, vendar omenja, da je zaenkrat še ni zasledil v praksi).

Model AMM je predstavljen v Tabeli 11.

Tabela 11: Agilne zrelostne stopnje (AMM)

Stopnja	**Tim**	**Oddelek**	**Poslovna stran**	**Projektno vodstvo**	**Poslovodstvo**	**Rezultat**
1	**2**	**3**	**4**	**5**	**6**	**7**
Prva ali skupinska	- imenovan je *Skrbnik Metodologije Scrum* - nedosledna uporaba nekaterih praks	brez sprememb	brez sprememb	brez sprememb	brez sprememb	- manjše izboljšanje produktivnosti agilnega tima
Druga ali oddelčna	- imenovan je *Skrbnik Metodologije Scrum* - dosledna uporaba nekaterih praks	- agilnost je občasno tema sestankov - več timov uporablja metodologija Scrum	- poslovni analitik deluje kot predstavnik poslovne strani	- se zavedajo, da nekateri timi uporabljajo agilne metode	brez sprememb	- manjše izboljšanje produktivnosti agilnih timov - nekateri timi so se vrnili na tradicionalni pristop
Tetja ali poslovna	- imenovan je *Skrbnik Metodologije Scrum*	- različno pojmovanje vlog poslovnega	- imenovan je *Produktni vodja*	- se zavedajo, da nekateri timi uporabljajo	- se zavedajo, da nekateri timi uporabljajo agilne metode	- manjše izboljšanje produktivnosti agilnih timov

Stopnja	Tim	Oddelek	Poslovna stran	Projektno vodstvo	Poslovodstvo	Rezultat
1	2	3	4	5	6	7
	- sodelovanje zunanjih svetovalcev	analitika, poslovnega arhitekta itn. - sodelovanje zunanjih svetovalcev		agilne metode	- obstaja »evangelist« oziroma sponzor, ki promovira agilnost	- izboljšanje morale agilnih timov - nekateri timi so se vrnili na tradicionalni pristop
Četrta ali projektna	- jasna ločitev med vlogami *Skrbnik Metodologije Scrum* in projektni vodja - dosledna uporaba metodologije Scrum	- dosledna uporaba večine elementov metodologije *Scrum* - usklajeno pojmovanje vlog poslovnega analitika, poslovnega arhitekta itn.	- imenovan je *Produktni vodja* - projektni vodja je del tima *Produktnega vodje*	- vključujejo agilne pristope v projektno vodenje	- se zaveda, da nekateri timi uporabljajo agilne metode - obstaja »evangelist« oziroma sponzor, ki promovira agilnost - izvaja se izobraževanje	- večja produktivnost agilnih timov - višja morala agilnih timov - manj nesoglasij glede delitve vlog
Peta ali poslovodna	- imenovan je *Skrbnik Metodologije Scrum* - jasna razdelitev Scrum vlog	- dosledna uporaba večine elementov metodologije Scrum - jasna razdelitev Scrum vlog	- imenovan je *Produktni vodja* - projektni vodja je del tima *Produktnega vodje*	- vključujejo agilne pristope v projektno vodenje	- omogoča timu, da deluje samostojno in odgovorno - prilagaja stil vodenja agilnim pristopom	- večja produktivnost agilnih timov - višja morala agilnih timov - organizacija se hitreje odziva na spremembe

Vir: po [Pro10c]

Končni rezultat pete ali poslovodne stopnje je, da se organizacija hitreje odziva na spremembe, agilnost pa je uvedena na nivoju tima, oddelka, poslovne strani, projektnega vodenja in poslovodenja. Pomembna značilnost pete ali poslovodne stopnje je, da je poslovodstvo prilagodilo stil vodenja agilnim pristopom, vključno z organizacijo in sistemom poročanja.

Scott Ambler je prvega aprila 2010 v e-reviji Dr. Dobbs [Amb10] opisal svojo različico zrelostnega modela agilnih metod (ang. AMM – Agile Maturity Model), v kateri so na duhovit način predstavljene stopnje prehoda iz tradicionalnih na agilne metode:

- Prva ali retorična stopnja (ang. Rhetorical stage), za katero je značilna nekritična navdušenost nad agilnimi metodami in pomanjkanje upoštevanja tradicionalnih metod in izkušenj;
- Druga ali certificirana stopnja (ang. Certified stage), v kateri je poudarek na pridobivanju certifikatov, pri čemer ni nujno, da certifikati odražajo razumevanje in poznavanje agilnih metod;
- Tretja ali verodostojna stopnja (ang. Plausible stage), za katero je značilno, da so razvijalci usmerjeni predvsem na kakovost programske opreme in manj na promocijo agilnosti oziroma pridobivanje certifikatov;

- Četrta ali ugledna stopnja (ang. Respectable stage), v kateri so razvijalci usmerjeni predvsem na kakovost rešitve, vključno s programsko opremo in pri svojem delu upoštevajo tudi izkušnje tradicionalnega razvoja programske opreme;
- Peta ali izmerljiva stopnja (ang. Measured stage), ko so zagotovljeni podatki, ki omogočajo poslovodstvu, da spremlja proces in sprejema ustrezne odločitve, pri čemer se upoštevajo izkušnje tako iz agilnega kot tudi tradicionalnega razvoja programske opreme.

Za prvi dve stopnji je značilen nekritičen pristop k agilnemu razvoju programske opreme in negiranje izkušenj, pridobljenih z uporabo tradicionalnih metod razvoja. Po začetnem navdušenju in zagonu pa je v nadaljnjih stopnjah prisoten hibriden pristop k razvoju programske opreme, ki vključuje izkušnje iz agilno in tradicionalno vodenih razvojnih projektov.

V obeh zgoraj opisanih agilnih zrelostnih modelih je za najvišjo stopnjo značilna vključenost poslovodstva in prilagoditev poslovodenja agilnim pristopom.

3.2.2 *Scrum* zrelostni modeli

Predhodni opis metodologije *Scrum* se nanaša na izvorno različico metodologije *Scrum*, kot je zapisana v [Sch02]. V praksi je v uporabi veliko inačic, ki se razlikujejo od izvorne. Jeff Sutherland, eden od avtorjev metodologije *Scrum,* je razvil preizkus, ki ga je poimenoval »ScrumToda« (ang. Scrumbut test«) in je ime dobil po stavku, s katerim so uporabniki pojasnjevali odstopanja od izvorne različice metodologije *Scrum*: »Uporabljamo metodologija *Scrum*, toda ...«, nakar je sledilo naštevanje odstopanj od izvorne različice metodologije *Scrum* [Sut09]. Preizkus omogoča preverjanje skladnosti z izvorno različico metodologije *Scrum*. Sestavljen je iz vprašanj, navedenih v Tabeli 12.

Tabela 12: Scrum preizkus – Scrumbut, Scrumbutt, Nokia preizkus

Oznaka	Odgovori	Število točk
1	***Iteracije***	
1-1	Brez *Iteracij*	0
1-2	Dolžina *Iteracije* > 6 tednov	1
1-3	Različne dolžine *Iteracij* < 6 tednov	2
1-4	Nespremenljiva dolžina *Iteracij* 6 tednov	3
1-5	Nespremenljiva dolžina *Iteracij* 5 tednov	4
1-6	Nespremenljiva dolžina *Iteracij* <= 4 tedne	10
2	**Testiranje**	
2-1	Ni zagotavljanja kakovosti (QA)	0
2-2	Testiranje enot (unit tested)	1
2-3	Testiranje funkcionalnosti (feature tested)	5
2-4	Testiranje funkcionalnosti takoj po izdelavi (feature tested as soon as completed)	7
2-5	Prevzemni test je uspešen	8
2-6	Programska oprema je v produkciji	10
3	**Agilne specifikacije**	
3-1	Ni *Zahtev*	0
3-2	Obsežno dokumentirane *Zahteve*	1
3-3	Nekakovostne *Uporabniške zgodbe*	4
3-4	Kakovostne *Zahteve*	5
3-5	Kakovostne *Uporabniške zgodbe*	7
3-6	Minimalne in pravočasne *Zahteve* (ang. »just enough, just in time«)	8
3-7	Kakovostne *Uporabniške zgodbe*, po potrebi povezane s specifikacijami	10
4	**Produktni vodja**	

Oznaka	Odgovori	Število točk
4-1	Ni *Produktnega vodje*	0
4-2	*Produktni vodja* ne razume metodologije *Scrum*	1
4-3	*Produktni vodja* moti *Razvojno skupino*	2
4-4	*Produktni vodja* ne sodeluje z *Razvojno skupino*	2
4-5	*Produktni vodja* pridobi ocene *Razvojne skupine* za *Seznam Uporabniških zgodb* pred *Načrtovalnim sestankom Iteracije*	5
4-6	*Produktni vodja* določa datume *Izdaje* glede na *Hitrost Razvojne skupine*	8
4-7	*Produktni vodja* motivira *Razvojno skupino*	10
5	**Seznam Uporabniških zgodb**	
5-1	Ni *Seznama Uporabniških zgodb*	0
5-2	Več *Seznamov Uporabniških zgodb*	1
5-3	En *Seznam Uporabniških zgodb*	3
5-4	Jasno opredeljen *Seznam Uporabniških zgodb,* s prioritetami določenimi glede na ROI pred *Načrtovalnim sestankom Iteracije*	5
5-5	*Produktni vodja* določa datume *Izdaje* glede na *Seznam Uporabniških zgodb*	7
5-6	*Produktni vodja* lahko izmeri ROI glede na dejanske prihodke, stroške na *točko Uporabniške zgodbe* ali druge metrike	10
6	**Ocene**	
6-1	*Seznam Uporabniških zgodb* ne vsebuje ocen	0
6-2	*Razvojna skupina* ni sodelovala pri ocenjevanju	1
6-3	Ocene so bile določene brez uporabe *Načrtovalnega pokerja*	5
6-4	Ocene je pripravila *Razvojna skupina* z uporabo *Načrtovalnega pokerja*	8
6-5	Napaka pri ocenjevanju je manjša od 10%	10
7	***Padajoči diagram spremljanja napredka***	
7-1	Ni *Padajočega diagrama napredka*	0
7-2	*Razvojna skupina* ne sodeluje pri posodabljanju *Padajočega diagrama napredka*	1
7-3	*Padajoči diagram spremljanja napredka* vključujejo tudi nedokončane *Naloge*	2
7-4	*Padajoči diagram spremljanja napredka* se zmanjšuje le, ko so *Naloge* dokončane	4
7-5	*Padajoči diagram spremljanja napredka* se zmanjšuje le, ko je *Uporabniška zgodba* potrjena	5
7-6	Dodati 3 točke, če je *Razvojna skupina* izračunala svojo *Hitrost*	+3
7-7	Dodati 2 točki, če je *Produktni vodja* določil datume *Izdaje* glede na znano *Hitrost*	+2

Vir: [Sut09]

Uspešnost navedenega Nokia preizkusa vpliva na znesek, ki so ga investitorji pripravljeni vložiti v podjetja za razvoj programske opreme [Sut09b]. Investitorje zanimajo predvsem podjetja, ki so hitrejša in bolj uspešna od povprečja.

Iz Tabele 12 je razvidno, da Nokia preizkus predvideva največje število točk za *Scrum* projekte, pri katerih se uporabljajo naslednje *Scrum* metrike:

- 6-5: Napaka pri ocenjevanju je manjša od 10%;
- 7-5: Padajoči diagram spremljanja napredka se zmanjšuje le, ko je *Uporabniška zgodba* potrjena;
- 7-6: *Hitrost Razvojne skupine* in
- 7-7: Datum *Izdaje* je določen glede na znano *Hitrost.*

3.2.3 Lastnosti »hiperproduktivnosti«

Sutherland [Sut09a] je definiral pogoje, pod katerimi je možno z uporabo metodologije *Scrum* doseči »hiperproduktivnost« oziroma bistveno večjo produktivnost v primerjavi s tradicionalnimi metodami.

Opazil je, da uporaba metodologije *Scrum* v organizacijah, ki dosegajo peto zrelostno raven po CMMI, znižuje vse kategorije dela za skoraj 50% [Sut07]. V novejših virih [Sut09a] navaja, da najboljše *Scrum* skupine na svetu dosegajo povprečno 750% večjo *Hitrost* kot slapovne skupine [Sut09a], kot primere pa navaja podjetja v Združenih državah Amerike, Rusiji, Nizozemski in Indiji, vendar pa istočasno opozarja, da okoli 90% *Scrum* skupin ne dosega tako dramatičnih rezultatov. Navedeno pojasnjuje z neustreznimi implementacijami metodologije *Scrum*, ki potekajo tako, da posamezniki preberejo nekaj knjig, se udeležijo nekaj usposabljanj in začnejo z implementacijo metodologije *Scrum*, posledično pa uporabljajo hibridni *Scrum*, pri katerem manjkajo ključni elementi. Sutherland navaja rezultate neformalne raziskave, po kateri 50% *Scrum* skupin ni zmožno ob koncu *Iteracije* testirati razvito programsko opremo, kar ni skladno z načeli Agilnega manifesta.

Kot rešitev priporoča način, ki ga primerja s šok terapijo, pri katerem *Scrum* skupina v treh *Iteracijah* dosledno upošteva pravila izvorne različice metodologije *Scrum*, vodi jih pa izkušeni *Skrbnik Metodologije Scrum. Scrum* skupina lahko spremeni pravila le po treh *Iteracijah* in sicer le pod pogojem, da je dosegla 240 % povečanje *Hitrosti* in da je sposobna spremembo utemeljiti s poslovnimi razlogi. Na opisani način so v dveh podjetjih (MySpace v Kaliforniji, JayWay na Švedskem) pri najboljših *Scrum* skupinah dosegli tudi »hiperproduktivnost«, kar pomeni, da je *Hitrost* tima bila vsaj 400 % večja od povprečne *Hitrosti* slapovnega tima Pri tem so upoštevali naslednjih deset pravil:

1. Vsi udeleženci se morajo udeležiti usposabljanja o metodologiji *Scrum*;
2. *Seznam Uporabniških zgodb* mora biti pripravljen pred *Načrtovalnim sestankom Iteracije*;
3. Programska oprema mora biti zaključena do *Potrjevalnega sestanka Iteracije;*
4. Če obstaja delo, ki ga lahko opravi samo ena oseba, je potrebno takoj tej osebi določiti par;
5. Ni večopravilnosti;
6. Fizična *Scrum* plošča;
7. Kratke *Iteracije* (pogosto en teden);
8. Padajoči diagram spremljanja napredka prikazuje samo točke *Uporabniških zgodb*;
9. Prioriteto za vse (vključno s podporo) določa *Produktni vodja*;
10. Uporablja se služeči (ang. servant) način vodenja.

3.3 Določitev metrik

3.3.1 Uvod

Pomanjkljivo spremljanje učinkovitosti agilnega razvoja programske opreme je ena od pomanjkljivosti agilnih metodologij razvoja programske opreme ([McB00], [Ste03], [Boe04], [Mah07a]). V literaturi navajajo, da deležniki pogrešajo metrike glede predvidenega zaključka projekta [Sch05] ter, da obstaja potreba po boljšem merjenju vrednosti glede na stroške [Yap06], pri čemer je potrebno upoštevati, da metrike, ki ne upoštevajo načel agilnosti, lahko povzročijo ravno nasproten učinek [Har06]. Možne rešitve so podane v ([Mah07a], [Bar05]).

Izvorna različica metodologije *Scrum* [Sch04] vsebuje samo dve metriki: merjenje preostalega obsega dela na ravni *Iteracije* (v urah) in na ravni *Izdaje* (merska enota ni predpisana). Predvidena je uporaba *Padajočih diagram napredka*. Z uvedbo *Uporabniških zgodb* [Coh04] je bila vpeljana tudi nova metrika: število točk, ki pomenijo oceno zahtevnosti razvoja *Uporabniških zgodb.* S tem je bil omogočen izračun *Hitrosti Razvojne skupine*, ki je osnova za natančneje načrtovanje dela. Izvorne *Scrum* metrike so bile nadgrajene z dodatnimi metrikami ([Bar05], [Sul06], [Dam07]).

Naslednji dve načeli agilnega razvoja programske opreme se nanašata na metrike [Nic09]:

- N1: Naša največja prioriteta je zadovoljstvo stranke z zagotavljanjem zgodnje in neprekinjene dobave programske opreme, katere vrednost je pomembna za stranko.
- N7: Delujoča programska oprema je glavno merilo napredovanja.

Zaželene lastnosti metrik [Gri10] so, da imajo pozitiven Hawthornov učinek[8], so enostavne, pomembne za doseganje končnega cilja in usmerjene v prihodnost (npr. prevzete funkcionalnosti, ocena sponzorja, zadovoljstvo uporabnika ter čas cikla za odpravljanje napak). Agilna pravila o metrikah [Nic09] pravijo, da je potrebno meriti rezultate in ne aktivnosti, ter da je potrebno meriti samo tisto, kar je nujno potrebno in nič več kot to (minimalistični pristop).

Metrike lahko delimo na [Nic09]: informativne (kažejo dejansko stanje), diagnostične (ugotavljajo možnosti za izboljšave) in motivacijske (vplivajo na obnašanje), kar je bolj podrobno opisano v Tabeli 13.

Tabela 13: Agilne metrike

ZŠ	Metrika	Načelo	Informativni	Diagnostični	Motivacijski
1	RTF (Delujoča testirana funkcionalnost)	N7	Neposredno merilo dostavljenih rezultatov	Če je dalj časa nespremenjen, potem so možni problemi	Člani *Razvojne skupine* se trudijo, da povečajo RTF
2	Strogo finančna vrednost	N1	Neposredno merilo dostavljene finančne vrednosti	Trend nižanja finančne vrednosti je uporaben pri odločanju o nadaljevanju projekta	Člani *Razvojne skupine* želijo prispevati k uspehu podjetja, udeleženci spremljajo finančno vrednost *Izdaje*
3	Prislužena	N1	Neposredno merilo	Če trend ni oblike črke S,	Člani *Razvojne skupine* želijo

[8] Hawthornov učinek se nanaša na ugotovitev, da merjenje vpliva na rezultat meritve. Pri eksperimentu, v katerem so povečali osvetlitev delovnega prostora, so opazili, da so delavci delali več tudi potem, ko so osvetlitev zmanjšali – na povečano produktivnost je vplivalo predvsem to, da so delavci vedeli, da so opazovani.

ZŠ	Metrika	Načelo	Informativni	Diagnostični	Motivacijski
	poslovna vrednost		dostavljene poslovne vrednosti	potem so možni problemi glede določanja prioritet ali vrednosti	prispevati k uspehu podjetja, udeleženci spremljajo poslovno vrednost *Izdaje*
4	*Hitrost*	N1	Uporabna za oceno datuma zaključka ali oceno obsega	Če obstajajo vzorci glede spreminjanja *Hitrosti*, potem so možni problemi	Člani *Razvojne skupine* so ponosni na *Hitrost* in jo ohranjajo
5	*Padajoči diagram spremljanja napredka*	N1	Neposredno merilo preostalega dela		

Vir: po [Nic09]

Postopoma so se pojavile raziskave na področju oblikovanja modelov za merjenje učinkovitosti agilnega razvoja programske opreme. V [Mah07b] je podan sistematičen pristop k oblikovanju metrik v skladu z zahtevami CMMI, v [Mah08a] je prestavljen načrt repozitorija za hranjenje podatkov o meritvah. Ktata and Levesque [Kta10] sta opisala pristop za merjenje *Scrum* projektov, ki temelji na metodi Cilj-Vprašanje-Metrika (ang. Goal-Question-Metric). Meta model procesa *Scrum,* ki so ga razvili na Univerzi v Milanu [Dam07] z uporabo Object Management Group Meta-Object Facility (MOF), vključuje dva meta modela: enega za ustvarjanje modelov za opis razvojnega procesa in drugega za ustvarjanje modelov za opis ogrodja za merjenje, pri katerem je uporabljena GQM paradigma (Goal/Question/Metric paradigm). Oba modela povezuje raven sprožilcev (ang. trigger layer).

3.3.2 Merjenje *Hitrosti*

Hitrost je enaka številu doseženih točk *Uporabniških zgodb* v eni *Iteraciji* in je pomembna za določanje datuma *Izdaje.*.

Točka zgodbe je idealen delovni dan. Ocene števila točk zgodb so lahko 0.5, 1, 2, 3, 5, 8, 13, in 20, kot je predlagal Cohn [Coh04]. Pri začetnih ocenah, ko *Razvojna skupina* še nima izkušenj pri planiranju, je ocena *Hitrosti* lahko enaka šest ur ali en človek dan. Pričakuje se, da bo po vsaki *Iteraciji* ocena bolj realna.

Pri spremljanju točk *Uporabniških zgodb* je ključnega pomena opredelitev zaključka *Uporabniške zgodbe* ali kdaj je *Uporabniška zgodba* potrjena (ang. done). Problem nejasnega kriterija zaključka zgodbe je opisan v [Mah10b] in [Moe09]. Možna rešitev je, da *Produktni vodja* sprejme samo tiste *Uporabniške zgodbe*, ki so testirane in pripravljene za produkcijsko okolje.

Pri uporabi metrike *Hitrost* obstaja tveganje, da bo *Razvojna skupina* ocenila preostali obseg dela z višjo oceno, da bi doseglo želeno stanje, ko je dejanska *Hitrost* večja od načrtovane.

3.3.3 Merjenje prislužene vrednosti

Ena izmed ovir, zaradi katere poslovodstvo ima pomisleke pri uporabi agilnih metod je tudi odsotnost tradicionalnih metrik, kot je uporaba prislužene vrednosti ([Raw06]. Običajne *Scrum* metrike kot so *Hitrost* in *Padajoči diagram spremljanja napredkai*, posredujejo

informacijo o *Hitrosti*, ne pa tudi o stroških ali poslovni vrednosti projekta ([Raw10]. Osnovna vprašanja, ki jih poslovodstvo zastavlja so:

- Ali bo realizirano, kar je plačano?
- Ali je realizacija dovolj hitra?
- Koliko vrednosti je realizirano?

Pri projektnem vodenju se je uveljavila uporaba kazalnika prislužene vrednosti (ang. »earned value«), ki je kombinacija kazalnika tehnične uspešnosti (ne/uresničitev planiranega dela), časovne uspešnosti (nedoseganje/preseganje rokov) in stroškovne uspešnosti (pod/nad proračunom) ([PMI08a], [PMI08b]). Prislužena vrednost je v uporabi več kot 100 let. Ključne komponente izračuna prislužene vrednosti so predstavljene v Tabeli 14.

Tabela 14. Ključne komponente izračuna prislužene vrednosti

Oznaka	Kratki opis	Daljši opis
PV	Planirana vrednost (ang. Planned value)	Planirano fizično delo in odobreni planirani stroški za izvedbo planiranega dela.
AC	Dejanski stroški (ang. Actual cost)	Povzročeni celotni stroški, ki se morajo navezovati na pripadajoče predračunane stroške pri planirani vrednosti in prisluženi vrednosti (to so včasih lahko samo neposredne delovne ure, samo neposredni stroški ali vsi stroški, vključujoč tudi posredne stroške), ki so nastali pri uresničevanju delovnih nalog - aktivnosti v danem časovnem obdobju. ↑ tudi prislužena vrednost.
EV	Prislužena vrednost (ang. Earned value)	Uresničeno, opravljeno fizično delo s potrjenim predračunom stroškov za opravljeno delo; vsota potrjenih ocen stroškov (ki lahko vsebuje tudi razporeditve splošnih stroškov) za aktivnosti (ali dele aktivnosti), ki so bile končane v danem obdobju (običajno na opazovani datum projekta).
BAC	Planirani stroški (ang. Budget at complete)	Vsota celotnih predračunskih stroškov projekta.
SPI	Terminski indeks (ang. Schedule Performance Index)	Razmerje (uspešnosti) učinkovitosti med prisluženo vrednostjo (EV) in planirano vrednostjo (PV). SPI opisuje, kolikšen delež planiranega terminskega plana je bil dosežen oz. realiziran. SPI = EV/PV.
CPI	Stroškovni indeks (ang. Cost Performance Index)	Razmerje stroškovne uspešnosti (učinkovitosti) med prisluženo vrednostjo (EV) in dejanskimi stroški (AC); CPI pogosto uporabljamo za napovedovanje velikosti možne prekoračitve stroškov; uporabljamo naslednjo formulo: BAC/CPI = napovedani stroški na koncu; CPI = EV/AC.
BCWP	Planirani stroški za opravljeno delo (ang. Budgeted Cost of Work Performed)	Pojem je nadomeščen s pojmom prislužena vrednost.
BCWS	Planirani stroški za planirano delo (ang. Budgeted Cost of Work Scheduled)	Pojem je nadomeščen s pojmom planirana vrednost.
ACWP	Dejanski stroški opravljenega dela (ang. Actual Cost of Work Performed)	Dejanski stroški
CV	Stroškovni odmik (ang. Cost Variance)	CV = EV - ACWP (oz. AC).
SV	Odmik terminskega plana (ang. Schedule Variance)	SV = EV - BCWS (PV)

Vir: [PMI08a]

Za projekte v ameriški javni upravi je uporaba kazalnika prislužene vrednosti obvezna za projekte, katerih vrednost je večja ali enako 20 milijonov ameriških [GSA12], pri čemer je

način uporabe prislužene vrednosti predpisan s standardom ANSI EIA 748-A: »Earned Value Management System« [ANS98]. Pri projektih, ki jih financira vlada, je obvezno poročanje prislužene vrednosti. Izkušnje pri uporabi prislužene vrednosti pri agilnih vrednosti v okolju javne oprave so podane v [All03]:

- *Hitrost* je bila nadomeščena z metrikami prislužene vrednosti;
- ustvarjene so bile BCWP z uporabo zahtev po testiranju;
- vzpostavljena je bila BCWS na začetku vsake *Iteracije*;
- s pomočjo uporabe sistema za evidenco časa se je zajela ACWP;
- izračun CV in SV je bil izveden iz treh osnovnih metrik prislužene vrednosti;
- enako velja za EAC in ETC.

Primerjava med tradicionalno prisluženo vrednostjo (T-EVM) in agilno prisluženo vrednostj (A-EVM) je podana v ([Wu11]. Ključni dejavnih uspešnosti uporabe prislužene vrednosti v tradicionalnih projektih je kakovost načrta. Pri agilnih projektih se načrt pogosto spreminja, zato je planirana vrednost, izračunana na klasičen način, neuporabna. V [Cab06] je navedene uporaba treh mehanizmov za spremljanje agilnih projektov: *Padajoči diagram spremljanja napredka, Rastoči diagram spremljanja napredkan diagram* in *Kumulativni diagram poteka projekta:*

- *Padajoči diagram spremljanja napredka* prikazuje obseg potrjenih funkcionalnosti kot funkcijo časa. Avtorja ugotavljata, da je obseg potrjenih funkcionalnosti neposredno primerljiv s prisluženo vrednostjo.
- *Rastoči diagram spremljanja napredka* prikazuje obseg nepotrjenih funkcionalnosti kot funkcijo časa in je uporaben za oceno doseganja roka.
- *Kumulativni diagram poteka projekta* prikazuje podobno informacijo kot *Padajoči diagram spremljanja napredka,* le da vsebuje podatke za vse statuse funkcionalnosti (ne samo potrjene, temveč tudi npr. čakajoče, začete, načrtovane, kodirane).

Avtorja [Cab06] primerjata časovni indeks SPI z odstopanji, ugotovljenimi pri *Rastočih in padajočih diagramih spremljanja napredka.* Tovrstni diagrami praviloma ne posredujejo informacij o stroških. Avtorja predlagata, da se na istem *Diagramu spremljanja napredka* prikazujejo potrjene funkcionalnosti in načrtovani ter dejanski stroški.
Avtorja [Cab06] predlagata naslednjo formulo za izračun časovnega indeksa SPI: potrjene funkcionalnosti/načrtovane funkcionalnosti, željena vrednost je večja ali enaka ena. Za izračun stroškovnega indeksa CPI avtorja predlagata naslednjo formulo: načrtovani stroški/dejanski stroški, željena vrednost je manjša ali enaka ena. Avtorja [Cab06] svetujeta uporabo navedenih diagramov namesto tradicionalnega koncepta prislužene vrednosti. V prispevku [Kan07] je podan predlog rešitve izračuna prislužene vrednosti z uporabo količinsko prilagojenega proračuna (ang. Quantity Adjusted Budget, QAB), ki se sicer uporablja pri gradbenih projektih.

V prispevku [Sul06] je opisana prilagoditev kazalnikov SPI in CPI za spremljanje *Scrum* projektov. Agilna prislužena vrednost (v nadaljevanju AgileEVM) se izračunava na ravni *Izdaje* in meri napredek ob koncu *Iteracije*, ko so znani podatki o dejanski *Hitrosti Iteracije* ter dejanskih stroških *Iteracije*. Komponente izračuna AgileEVM so predstavljene v Tabeli 15.

Tabela 15: Ključne komponente izračuna agilne prislužene vrednosti

Oznaka	Kratki opis	Daljši opis
APC	Dejanski odstotek dokončanega dela I*zdaje* (ang. Actual Percent Complete of Release)	APC = Število točk potrjenih *Uporabniških zgodb* / Število točk načrtovanih *Uporabniških zgodb*
PPC	Planirani odstotek dokončanega dela *Izdaje* (ang. Planned percent complete)	PPC = Število zaključenih *Iteracij* /Število planiranih *Iteracij*
EV	Prislužena vrednost (ang. Earned value)	EV = APC * BAC
PV	Planirana vrednost (ang. Planned value)	PV = PPC * BAC

Vir: [Sul06]

Za izračun parametra APC v Tabeli 15 se uporabljajo točke *Uporabniških zgodb,* ki jih določijo člani *Razvojne skupine*. V prispevku ([Chr11] avtor predlaga, da izračunamo ceno točke *Uporabniške zgodbe* tako, da proračun *Iteracije* delimo s številom točk *Uporabniških zgodb*, predvidenih za to *Iteracijo* in potem izračunano ceno uporabimo pri izračunu CPI kot razmerju med predvidenim številom točk *Uporabniških zgodb,* izraženim v denarni enoti in številom točk potrjenih *Uporabniških zgodb*, izraženim v denarni enoti. [Rus09] predlaga uporabo BWCP oziroma EV pri *Padajočih diagramih spremljanja napredka* in primerja vrednost s številom točk *Uporabniških zgodb* ter spremljanje stroškov projekta z uporabo treh linij: sive, zelene in rdeče. Siva linija prikazuje predvideni napredek (PV oziroma BCWS), zelena linija dejanski napredek (EV ali BCWP) in rdeča linija dejanske stroške (AC ali ACWP), pri čemer so vse tri vrednosti izražene v odstotkih. Dejanske stroške izračunamo tako, da število porabljenih ur pomnožimo s ceno ure. Porabljene ure se evidentirajo tedensko, skupaj za vse naloge. Predpogoj za uspešno uporabo tovrstnega načina je opredelitev obsega na začetku *Izdaje* ter linearen potek razvoja.

Za razliko od do sedaj opisanih pristopov, kjer izračun prislužene vrednosti uporablja število točk *Uporabniških zgodb*, Rawsthorhne [Raw06] predlaga uporabo kazalnika prislužena poslovna vrednost ali EBV (ang. Earned Business Value), ki namesto števila točk *Uporabniških zgodb* uporablja utež, ki odraža poslovno vrednost. V [Raw10] je navedeno, da je pri izračunu prislužene poslovne vrednosti pomembna utež, ki je enaka poslovni vrednosti in ni nujno proporcionalna številu točk *Uporabniških zgodb,* ki odražajo zahtevnost dela. Npr. *Uporabniška zgodba* ima lahko visoko poslovno vrednost in posledično visoko utež, ter nizko stopnjo zahtevnosti in posledično majhno število točk. Avtor predlaga tudi uporabo uteži pri izbiri *Uporabniških zgodb*, ki bodo predmet razvoja v *Iteraciji*. Poleg kapacitete *Razvojne skupine*, ki je odvisna od *Hitrosti Razvojne skupine*, je pri oblikovanju *Seznama Nalog Iteracije* pomembno upoštevati tudi poslovno vrednost *Uporabniških zgodb*, ki zagotovlja ustrezno razmerje med stroški in realizirano poslovno vrednostjo *Iteracije.* Npr. z realizacijo manjšega števila *Uporabniških zgodb* z visoko poslovno vrednostjo lahko realiziramo večjo poslovno vrednost kot z večjim številom *Uporabniških zgodb* z nizko poslovno vrednostjo.

Vsi navedeni pristopi merijo prisluženo vrednost na ravni projekta oziroma *Izdaje*.

3.3.4 Ocenjevanje zahtevnosti *Uporabniških zgodb*

Za pravilno načrtovanje *projekta, Izdaje in Iteracije* so pomembne čim bolj točne ocene *Uporabniških zgodb.* Za njihovo ocenjevanje se pogosto uporablja *Načrtovalni poker,* ki je tudi predmet raziskav, ki ukvarjajo s točnostjo ocen, dobljenih po tej metodi, in vprašanjem, ali Načrtovalni poker odpravlja nagnjenost k optimizmu.

Točnost ocen se nanaša na razliko med načrtovano in dejansko zahtevnostjo *Uporabniških zgodb*, izraženo v točkah ali urah. Optimistične napovedi pomenijo zamude, zato je cilj ocenjevanja pridobiti čim bolj realne napovedi. Izvorna različica metodologije *Scrum* vključuje ocene preostalega dela, vendar pa ne predvideva poznejše analize ocen in primerjave ocen z dejanskimi podatki. V literaturi najdemo podatke o tem, da so napovedi strokovnjakov preveč optimistične [Jor04] ter da tehnika *Načrtovalnega pokerja* (ang. planning poker), ki se uporablja pri metodologiji *Scrum*, ne odstrani popolnoma pretiranega optimizma [Mol08].

Analiza točnosti ocen omogoča *Skrbniku Metodologije Scrum*, da ugotovi stopnjo optimizma ocen posameznih članov *Razvojne skupine* in prispeva k čim bolj realnim ocenam zahtevnosti *Uporabniških zgodb.* Za analizo točnosti ocen se uporablja pet različnih mer, pri čemer oznaka A pomeni dejanska zahtevnost *Uporabniške zgodbe*, oznaka E pa pomeni ocenjena zahtevnost *Uporabniške zgodbe.* Posamezno mero izračunamo s pomočjo formul (1) do (5).

- absolutna napaka (ang. absolute error):

$$AE = |A - E| \tag{1}$$

- relativna napaka (ang. relative error):

$$RE = \frac{|A - E|}{A} \tag{2}$$

- pristranska relativna napaka (ang. biased relative error):

$$REbias = \frac{(A - E)}{A} \tag{3}$$

- uravnoteženo merilo relativne napake (ang. balanced measure of relative error):

$$BRE = \frac{|A - E|}{\min(A, E)} \tag{4}$$

- pristransko uravnoteženo merilo relativne napake (ang. biased balanced measure of relative error):

$$BREbias = \frac{(A - E)}{\min(A, E)} \tag{5}$$

Pri merah *REbias* in *BREbias* je razvidna tudi smer napake (pozitivna pomeni optimistično oceno, negativna pa pesimistično oceno).

3.4 Poročanje o napredku

3.4.1 Uvod

Uporaba kazalnikov je tesno povezana z zbiranjem podatkov in prikazovanjem. V literaturi [Mah07a] je podan predlog, da se pri spremljanju kazalnikov prouči možnost uporabe dobre prakse na področju poslovne inteligence, ki zajema zbiranje relevantnih podatkov v podatkovnem skladišču, sprotno opozarjanje uporabnika na odstopanja od še sprejemljivih vrednosti ter prikaz v obliki nadzornih plošč.

3.4.2 Prikaz *Padajočega diagrama napredka*

Prvotno se je pri metodologiji *Scrum* za spremljanje učinkovitosti uporabljal samo *Padajoči diagram spremljanja napredka Izdaje*, ki prikazuje obseg preostalega dela na začetku vsake *Iteracije*, ki je izražen kot seštevek točk vseh nepotrjenih *Uporabniških zgodb* v *Seznamu Uporabniških zgodb*. Trend diagrama prikazuje verjetnost doseganja roka *Izdaje*.

V organizacijah, ki so na petem zrelostnem nivoju po CMMI [Jak08], se napredek *Scrum* projekta na nivoju *Iteracije* spremlja primarno z uporabo *Padajočih diagram napredka* in *Potrjevalnih sestankov Iteracije.* Navedeni diagram omogoča sprotno spremljanje dejanskega poteka projekta glede na pričakovano stanje (npr. situacijo, ko se obseg preostalega dela linearno zmanjšuje sorazmerno s časom, ki je pretekel od začetka projekta). Če je potek projekta hitrejši od planiranega, potem je dejanska krivulja pod idealno, v primeru, ko je projekt počasnejši od planiranega, pa je dejanska krivulja nad idealno.

3.4.3 Prikaz *Linije ravnotežja (LOB)*

Linija ravnotežja (ang. Line of Balance, LOB) je eden izmed načinov poročanja o napredku pri *Scrum* projektih [Mir10]. Avtorja ugotavljata, da je zelo malo standardiziranih mehanizmov za spremljanje in poročanje projektov, z izjemo prislužene vrednosti, ki je skoraj obvezna pri vseh večjih projektih, ki jih financira država ter, da različne agilne skupnosti uporabljajo različne prakse.

Scrum skupnost uporabljajo *Padajoče diagram napredka*, FDD (ang. feature driven development) skupnost uporablja kumulativne diagrame, XP (ang. extreme programming) skupnost pa uporablja število potrjenih *Uporabniških zgodb*. Avtorja ugotavljata, da so navedeni načini enostavni in razumljivi, vendar pa ne sporočajo celovite informacije o statusu projekta. *Padajoči diagram spremljanja napredkai* in število potrjenih *Uporabniških zgodb* posredujejo informacijo o obsegu preostalega dela in načrtovanem poteku projekta, ne pa tudi informacije o nepotrjenih *Uporabniških zgodbah*. Kumulativni diagrami posredujejo informacijo o nepotrjenih *Uporabniških zgodbah*, ne pa tudi o načrtovanem obsegu potrjenih *Uporabniških zgodb*. *Linija ravnotežja* podaja informacije, ki jih ne podaja *Padajoči diagram spremljanja napredka,* ki se uporablja pri *Scrum* projektih ali kumulativni diagram, ki se uporablja pri FDD projektih.

3.4.4 Prikaz plošč

Plošča nalog vsebuje *Naloge*, ki se nanašajo na posamezno *Uporabniško zgodbo*, status posamezne *Naloge*, teste, ki se nanašajo na posamezno *Uporabniško zgodbo* in drugo. Statusi se označijo z uporabo semaforja, kar prispeva k transparentnemu spremljanju napredka. Objava na web strani, wiki ali v orodju za poročanje dodatno prispeva k transparentnosti spremljanja.

Uporaba dobre prakse pri prikazovanju podatkov [Wat07] omogoča hitre odgovore na vprašanja, ki se nanašajo na preteklost (»Kaj se je zgodilo?«, »Zakaj se je to zgodilo?«) in prihodnost (»Kaj se bo zgodilo?«). Način prikazovanja podatkov lahko bistveno vpliva na pravilno interpretacijo kazalnikov in posledično na uspeh uporabe kazalnikov. Pri upravljanju učinkovitosti poslovanja se za prikaz kazalnikov uporabljajo nadzorne plošče ([Eck06], [Gol04]), na katerih so na prijazen in enostaven način prikazane trenutne vrednosti ključnih kazalnikov, ciljne vrednosti in trendi glede na predhodna obdobja.

Pri *Scrum* projektih se za spremljanje projekta pogosto uporabljajo *Scrum* plošče ([Sut07], [Gup08]). *Razvojne skupine* na *Scrum* ploščah objavljajo *Padajoče diagram napredka*, sezname *Ovir* in njihovih prioritet, *Naloge*, ki se nanašajo na posamezno *Uporabniško zgodbo*, status posamezne *Naloge*, teste, ki se nanašajo na posamezno *Uporabniško zgodbo* in drugo. Z uporabo semaforja označijo statuse in tako zagotovijo transparentno spremljanje napredka. Objava na web strani, wiki ali v orodju za poročanje dodatno prispeva k transparentnosti spremljanja. Uporaba fizičnih, enostavnih plošč je ena od zahtev tehnike »šok terapije« za uvedbo metodologije *Scrum* [Sut09a]. *Scrum* plošče imajo lahko različno vsebino, praviloma pa vsebujejo:

- *Padajoči diagram spremljanja napredka*;
- seznam *Ovir* s prioritetami;
- tri stolpce z različnimi statusi *Uporabniških zgodb* (v pripravi, v delu, potrjene);
- nenačrtovane *Uporabniške zgodbe* in *Uporabniške zgodbe*, ki bodo prenesene v naslednjo *Iteracijo;*
- *Razvojne skupine* uporabljajo *Scrum* plošče, ki so ponavadi fizične ([Sut07], [Gup08]), lahko pa so tudi elektronske in objavljene na intranetu.

Ena izmed možnih rešitev spremljanja *Scrum* projektov s pomočjo nadzorne plošče [Bar05] vsebuje prikaz kazalnikov z vidika poslovodstva, kjer so na eni strani grafično predstavljeni statusi napredka za posamezne dele izdelka (pri tem so uporabili poročanje iz metode FDD, shemo statusov pa so poimenovali »Parking Lot«), razmerje med planiranim in izvedenim delom (»Product Burndown Chart«), razmerje med planirano in doseženo poslovno vrednostjo (»Business Value Chart«) ter diagram strukturirane členitve nalog (»Work Breakdown Structure«).
Uporaba nadzorne plošče omogoča udeležencem, da se lahko v vsakem trenutku seznanijo s statusom projekta. Udeležencem, ki imajo vlogo »pujs« plošča omogoča osnovo za prihodnje aktivnosti (koliko je še potrebno narediti do zaključka *Iteracije*), za udeležence, ki imajo vlogo »kura« pa plošča omogoča spremljanje preteklih aktivnosti (koliko je že narejeno od začetka *Iteracije*). Padajoči diagram spremljanja napredkai so zaradi svoje enostavnosti in sporočilnosti primerni za vlogo »kura«, seznami pa so bolj podrobni in zaradi tega bolj primerni za vlogo »pujs«.

Vendar pa *Scrum* plošče praviloma na vsebujejo podatkov o stroških, ki so za poslovodstvo pomembni in jih običajno spremlja z uporabo indeksa CPI ali EV.

3.5 Modeli za spremljanje učinkovitosti

3.5.1 Model PMBOK

Proces spremljanja učinkovitosti v standardu PMBOK Vodnik ali »Vodnik po znanju projektnega vodenja« (ang. Guide to the Project Management Body of Knowledge) [PMI08a] obsega 10 procesov, ki skupaj predstavljajo procesno skupino »Procesi spremljanja in nadzora«.

V Tabeli 16 so navedeni procesi spremljanja in nadzora, njihovi vhodi in izhodi.

Tabela 16: PMBOK: Procesi spremljanja in nadzora

Proces	Vhodi	Izhodi
Področje znanja: 4. Obvladovanje integracije		
4.4 Nadzor dela	Plan projekta Poročila o učinkovitosti Dejavniki iz okolja Procesni viri organizacije	Zahteve po spremembi Ažuriranje plana projekta Ažuriranje projektne dokumentacije
4.5 Nadzora integracije sprememb	Plan projekta Informacije o učinkovitosti dela Zahteve po spremembi Dejavniki iz okolja Procesni viri organizacije	Ažuriranje zahtev po spremembi Ažuriranje plana projekta Ažuriranje projektne dokumentacije
Področje znanja: 5. Obvladovanje obsega		
5.4 Potrditev obsega	Plan projekta Dokumentacija o zahtevah Matrika sledljivosti zahtev Potrjeni izdelki	Sprejeti izdelki Zahteve po spremembi Ažuriranje plana projekta
5.5 Nadzor obsega	Plan projekta Informacije o učinkovitosti dela Dokumentacija o zahtevah Matrika sledljivosti zahtev Procesni viri organizacije	Vrednosti meril učinkovitosti dela Ažuriranje procesnih virov organizacije Zahteve po spremembi Ažuriranje plana projekta Ažuriranje projektne dokumentacije
Področje znanja: 6. Obvladovanje časa		
6.6 Nadzor urnika aktivnosti	Plan projekta Terminski plan projekta Informacije o učinkovitosti dela Procesni viri organizacije	Merila učinkovitosti dela Ažuriranje procesnih virov organizacije Zahteve po spremembi Ažuriranje plana projekta Ažuriranje projektne dokumentacije
Področje znanja: 7. Obvladovanje stroškov		
7.3 Nadzor stroškov EVM Orodja	Plan projekta Zahteve po financiranju projekta Vrednosti meril učinkovitosti dela Procesni viri organizacije	Merila učinkovitosti dela (CV, SV, CPI, SPI) Napovedi proračuna (EAC) Ažuriranje procesnih virov organizacije Zahteve po spremembi Ažuriranje plana projekta Ažuriranje projektne dokumentacije
Področje znanja: 8. Obvladovanje kakovosti		
8.3 Nadzor kakovosti	Plan projekta	Rezultati merjenja kakovosti

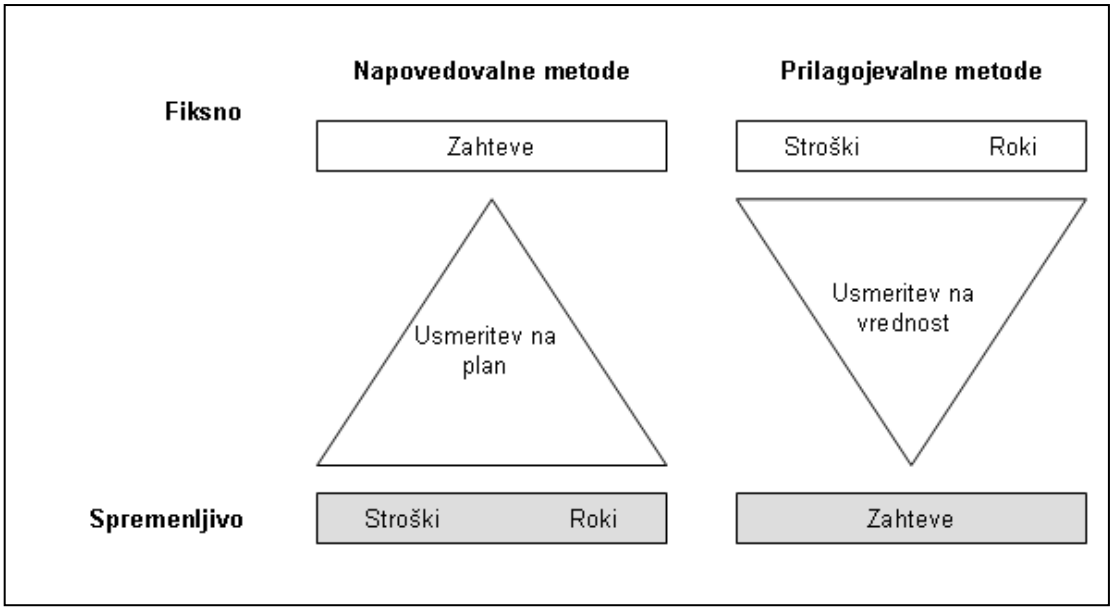
Proces	Vhodi	Izhodi
Kontrolni diagrami Orodja	Merila kakovosti Vprašalnik o kakovosti Vrednosti meril učinkovitosti dela (tehnična, časovna, stroškovna učinkovitost) Odobrene zahteve po spremembi Izdelki Procesni viri organizacije	Potrjene spremembe Potrjeni izdelki Ažuriranje procesnih virov organizacije Zahteve po spremembi Ažuriranje plana projekta Ažuriranje projektne dokumentacije
Področje znanja: 10. Obvladovanje komuniciranja		
10.5 Poročanje o doseganju ciljev Orodja	Plan projekta Informacije o učinkovitosti dela Vrednosti meril učinkovitosti dela Napovedi proračuna (EAC) Procesni viri organizacije	Poročila o učinkovitosti Ažuriranje procesnih virov organizacije Zahteve po spremembi
Področje znanja: 11. Obvladovanje tveganj		
11.6 Spremljanje in nadzor Orodja	Register tveganj Plan projekta Informacije o učinkovitosti dela Poročila o učinkovitosti	Ažuriranje registra tveganj Ažuriranje procesnih virov organizacije Zahteve po spremembi Ažuriranje plana projekta Ažuriranje projektne dokumentacije
Področje znanja: 12. Obvladovanje oskrbovanja		
12.3 Upravljanje oskrbovanja	Dokumenti oskrbovanja Plan projekta Pogodba Poročila o učinkovitosti Odobrene zahteve po spremembi Informacije o učinkovitosti dela	Dokumenti oskrbovanja Ažuriranje procesnih virov organizacije Zahteve po spremembi Ažuriranje plana projekta

Vir: [PMI08a]

Iz Tabele 14 je razvidno, da procesi spremljanja in nadzora vključujejo nadzor integracije sprememb, obsega, urnika aktivnosti, stroškov, kakovosti, tveganja in oskrbovanja. V modelu CMMI so pri opisu posebne prakse »SP 1.1 Spremljati planske parametre projekta« za procesno področje »PMC-Spremljanje in kontroliranje projekta« navedeni planski parametri oziroma tipični kazalniki napredovanja projekta in njegove učinkovitosti, ki vključujejo atribute rezultatov dela in nalog (npr. velikost, zahtevnost, raven storitve, razpoložljivost) ter stroške, vloženo delo in časovnico.

Navedeni nabor kazalnikov je pogosto poenostavljen na kombinacijo stroškov, rokov in zahtev projekta, ki je znana pod imenom »železni trikotnik« (ang. iron triangle) [Chi04]. V tem poglavju bo uporabljen »železni trikotnik« z namenom jasnejšega prikaza razlike med tradicionalnim in agilnim pristopom. Razlika med tradicionalnim in agilnim pristopom k projektnemu vodenju je, da so pri tradicionalnih pristopih fiksirane zahteve in se prilagajajo stroški in roki, pri agilnih pristopih pa so fiksirani stroški in roki in se spreminjajo zahteve, kot je prikazano na Sliki 15 ([Lac10a], [Hun09a]).

Slika 15: »Železni« trikotnik



Vir: po [Lac10a], [Hun09a]

3.5.2 Model SWEBOK

Proces spremljanja učinkovitosti je v »Vodniku po znanju programskega inženirstva SWEBOK« (ang. Software Engineering Body of Knowledge) [IEE08] obravnavan v okviru področja znanja »Proces programskega inženirstva« (ang. »Software engineering process«) in podpodročja »Merjenje procesa in izdelka«.

Pri obravnavi modelov za ocenjevanje procesov je v SWEBOK naveden model CMMI, katerega splošni opis je bil podan v drugem poglavju doktorske disertaacije. Proces spremljanja učinkovitosti je v najnovejši različici modela CMMI [CMM10] za razvojne procese z oznako CMMI-DEV V1.3, obravnavan pri splošnem cilju »GG 2 Uveljaviti obvladovan proces« (druga raven zmožnosti) in splošni praksi »GP 2.8 Spremljati in kontrolirati proces« ter pri posebnih ciljih procesnih področij »MA-Merjenje in analiza« (druga raven zrelosti), »OPP-Učinkovitost procesov organizacije« (četrta raven zrelosti) in »PMC-Spremljanje in kontroliranje projekta« (druga raven zrelosti).

Splošna praksa »GP 2.8 Spremljati in kontrolirati proces« se nanaša na spremljanje in kontroliranje procesa glede na plan izvedbe procesa ter izvajanje ustreznih korektivnih aktivnosti. Namen splošne prakse GP 2.8 je izvedba neposrednega dnevnega spremljanja in kontroliranja procesa, da bi se zagotovila ustrezna transparentnost in da bi se po potrebi lahko sprožile ustrezne aktivnosti. Spremljanje in kontroliranje procesa lahko vključuje merjenje ustreznih atributov procesa ali izdelkov, ki so rezultat procesa, kar je podrobneje opisano pri procesnih področjih »MA-Merjenje in analiza«, »OPP-Učinkovitost procesov organizacije« in »PMC-Spremljanje in kontroliranje projekta«. V Tabeli 17 so navedeni izbrani posebni cilji in posebne prakse za spremljanje učinkovitosti.

Tabela 17: Posebni cilji in prakse za spremljanje učinkovitosti

Posebni cilji (oznaka SG)	Posebne prakse (oznaka SP)
Procesno področje »MA-Merjenje in analiza«	
SG 1 Uskladiti aktivnosti merjenja in analize	SP 1.1 Opredeliti meritvene cilje
	SP 1.2 Opredeliti merila
	SP 1.3 Opredeliti postopke za zbiranje in shranjevanje podatkov
	SP 1.4 Opredeliti postopke za analizo
SG 2 Predstaviti rezultate merjenja	SP 2.1 Pridobiti meritvene podatke
	SP 2.2 Analizirati meritvene podatke
	SP 2.3 Shraniti podatke in rezultate
	SP 2.4 Sporočiti rezultate
Procesno področje »OPP-Učinkovitost procesov organizacije«	
SG 1 Vpeljati osnovne vrednosti in modele učinkovitosti	SP 1.1 Vpeljati cilje kakovosti in učinkovitosti procesa
	SP 1.2 Izbrati procese
	SP 1.3 Vpeljati merila učinkovitosti procesa
	SP 1.4 Analizirati učinkovitost procesa in določiti osnovne vrednosti učinkovitosti procesa
	SP 1.5 Vpeljati modele učinkovitosti procesov
Procesno področje »PMC-Spremljanje in kontroliranje projekta«	
SG 1 Spremljati projekt glede na načrt	SP 1.1 Spremljati planske parametre projekta
	SP 1.2 Spremljati zavezanost
	SP 1.3 Spremljati projektna tveganja
	SP 1.4 Spremljati obvladovanje podatkov
	SP 1.5 Spremljati sodelovanje deležnikov
	SP 1.6 Izvesti pregled napredovanja
	SP 1.7 Izvesti pregled mejnika

Vir: [CMMI10]

Pri razvoju programske opreme merimo tri vrste objektov [Rož03]: procese, izdelke in vire, pri katerih lahko glede na vpliv okolja merimo notranje atribute, na katere okolje nima vpliva in zunanje atribute, na katere ima okolje velik vpliv. Nekateri notranji in zunanji atributi razvoja programske opreme so predstavljeni v Tabeli 18.

Tabela 18: Preglednica notranjih in zunanjih atributov pri razvoju programske opreme

ZŠ	Objekt	Notranji atributi	Zunanji atributi
1	Procesi	Trajanje Obseg dela Število napak	Stroški
2	Izdelki	Velikost Strukturiranost Zahtevnost Funkcionalnost	Kakovost Razumljivost Uporabnost Primernost za vzdrževanje
3	Viri	Velikost Cena Strukturiranost	Kakovost Produktivnost Zanesljivost Uporabnost

Vir: po [Rož03]

3.5.3 Sistem PPMS

Sistem za merjenje učinkovitosti procesa (ang. process performance measurement system ali PPMS) [Kue00] opredeljuje učinkovitost procesa kot stopnjo doseganja ciljev vseh deležnikov pri procesu in zagotavljanja nenehnega izboljševanja procesa. Pregled literature na področju PPMS ([Mil11] je pokazal, da na tem področju obstaja pomanjkanje raziskav ter da obstoječi sistemi za merjenje učinkovitosti niso procesno usmerjeni.

Cilj sistema PPMS je zagotoviti celovito in pravočasno informacijo o učinkovitosti procesa. Tovrstna informacija omogoča komuniciranje ciljev in učinkovitosti procesa neposredno procesnemu timu, izboljšanje alokacije virov in rezultatov procesa glede količine in kakovosti, sporočanje zgodnjih opozorilnih znakov, izvajanje diagnoze pomanjkljivosti procesa, odločanje glede potrebe po korektivnih ukrepih in ocenjevanje posledic izvedenih ukrepov. Lastnosti sistema PPMS so [Kue00]:

1. zbiranje podatkov o procesih z uporabo kazalnikov učinkovitosti;
2. primerjava trenutnih in zgodovinskih ter ciljnih vrednosti kazalnikov in
3. sporočanje rezultatov (trenutna vrednost, ciljna vrednost, razkorak in trend za posamezni kazalnik) deležnikom procesa.

Možni pristopi k merjenju učinkovitosti so podani na Sliki 16. Pristop PPMS je usmerjen na poslovne procese in zajema kvalitativni ter kvanitativni vidik.

Slika 16: Pristopi k merjenju učinkovitosti

	Usmerjenost na poslovne enote	Usmerjenost na poslovne procese
Kvalitativni in kvantitativni vidik	Sistem uravnoteženih kazalnikov (ang. Balance Scorecard, BSC)	Merjenje učinkovitosti procesa (ang. Process Performance Measurement System, PPMS)
Kvantitativni vidik	Tradicionalna kontrola	Sistem razporejanja stroškov na podlagi aktivnosti (ang. Activity Based Costing, ABC)

Vir: [Kue00]

Koraki za vzpostavitev sistema PPMS (ang. Process Performance Measurement System) so:

1. opredelitev procesa;
2. opredelitev deležnikov procesa;
3. opredelitev ciljev deležnikov procesa;
4. opredelitev kazalnikov za doseganje ciljev deležnikov procesa;
5. opredelitev ciljnih in mejnih vrednosti kazalnikov procesa;
6. opredelitev izračuna kazalnikov procesa.

Najprej je potrebno opredeliti proces in njegove deležnike. Skladno s konceptom sistema PPMS so osnovni deležniki: vlagatelji, zaposleni, stranke in družba. Vsaka skupina deležnikov je zastopana z enim vidikom ali dimenzijo učinkovitosti: finančni vidik (za merjenje zadovoljstva vlagateljev), vidik zaposlenih, vidik strank in družbeni vidik. Poleg navedenih vidikov je za dolgoročno zadovoljstvo vseh štirih deležnikov pomemben vidik inovacije oziroma stalnih izboljšav. V naslednjem koraku je potrebno za vsakega deležnika proces določiti cilje, nato pa še kazalnike za doseganje ciljev deležnikov procesa. Pri tem je pomembno, da so kazalniki kvantitativni in kvalitativni. Za vsakega izmed kazalnikov je potrebno določiti ciljne in mejne vrednosti ter opredeliti izračun kazalnikov.

Koncept sistema PPMS smo uporabili kot izhodišče pri oblikovanju našega modela, kar smo podrobneje predstavili v naslednjem poglavju, kjer smo sledili zgoraj navedenim korakom. V Kuengov model smo vpeljali metrike z namenom zagotoviti zadovoljstvo vseh deležnikov ter pri tem ohraniti agilnost.

4. MODEL ZA SPREMLJANJE UČINKOVITOSTI AGILNEGA RAZVOJA PROGRAMSKE OPREME (*AGIT*)

4.1 Razvoj modela *AGIT*

4.1.1 Uvod

V tem poglavju bomo predstavili model za spremljanje učinkovitosti agilnega razvoja programske opreme, ki smo poimenovali *AGIT*. Razvoj modela *AGIT* je potekal postopoma, v okviru raziskav Laboratorija za tehnologijo programske opreme na Fakulteti za računalništvo in informatiko Univerze v Ljubljani na področju agilnih metodologij razvoja programske opreme in na področju merjenja programske opreme.

V Laboratoriju za tehnologijo programske opreme je bilo metodologija *Scrum* prvič uporabljeno pri razvoju aplikativne programske opreme za področje študijske informatike leta 2005, v okviru projekta »e-študent« [Mah05]. Po tej metodi je laboratorij razvil modul za vzdrževanje, ki vključuje vzdrževanje šifrantov, predmetnikov in podatkov o izvajalcih posameznih predmetov, administracijo uporabnikov, izvoz podatkov v podatkovno skladišče, uvoz podatkov o novih študentih in podobno. Dolžina *Iteracije* je bila 30 koledarskih dni oziroma 20 delovnih dni. Namesto *Uporabniških zgodb* so bili uporabljeni opisi *Zahtev* in *Nalog*, katerih zahtevnost je bila ocenjena v urah. Vse zahteve v *Seznamu Uporabniških zgodb* so bile uresničene. Merjenje učinkovitosti je vključevalo spremljanje preostalega časa ter statusov *Nalog*.

Na podlagi pridobljenih izkušenj je bila leta 2007 razvita prva verzija modela za spremljanje učinkovitosti razvoja programske opreme z uporabo metodologije *Scrum* [Mah07a], ki je temeljila na konceptu sistema za merjenje učinkovitosti procesa (ang. process performance measurement system), opisanega v [Kue00]. V modelu so bili upoštevani vidiki treh interesnih skupin ali deležnikov v razvojnem procesu: vodstva IT, članov *Razvojne skupine* in uporabnikov, kar zagotavlja uravnotežen pristop. Za vsakega deležnika so bili določeni cilji, nato pa izbrani ustrezni kazalniki za spremljanje doseganja posameznih ciljev. Vrednost kazalnika je bila izračunana s pomočjo metrik, katerih vrednosti so se zbirale med izvajanjem procesa. V modelu je bil predstavljen tudi predlog izračuna prislužene vrednosti na dnevni ravni. Navedeni model je bil izhodišče za model, ki predstavlja rezultat disertacije in je opisan v tem poglavju. Dopolnjen model vsebuje spoznanja in izkušnje, ki so bile pridobljene v obdobju od objave prve verzije do danes.

4.1.2 Primerjava modela *AGIT* z drugimi modeli

Konec leta 2007 je bil opisani model nadgrajen z vidikom *Skrbnika Metodologije Scrum* ter načrtom podatkovnega modela, upoštevajoč priporočila modela CMMI [Mah07b]. V sklopu nadaljevanja raziskovalnega dela smo leta 2008 izvedli primerjavo modela in zahtev v predhodnih poglavjih opisanega ogrodja COBIT [ITG07a], ki se uporablja pri reviziji informacijskih sistemov. Takrat smo tudi sprejeli odločitev, da naš model poimenujemo *AGIT* (AGIlna Informacijska Tehnologija). Ugotovili smo, da model *AGIT* vključuje večji del

ključnih COBIT indikatorjev, manjkajoče indikatorje pa smo vključili v predlagano nagradnjo modela. [Mah08c].

Konec leta 2011 smo revidiranje uspešnosti in učinkovitosti projekta informacijske tehnologije opisali v zaključnem delu za pridobitev naziva preizkušeni notranji revizor [Žab12]. Pri tem smo uporabili že omenjeni model COBIT [ITG07a] ter smernice za revidiranje projektov informacijske tehnologije GTAG 12 ki jih je izdal Inštitut notranjih revizorjev [IIA09]. Spoznanja pri izdelavi tega dela smo vključili v poglavje o ocenjevanju modela *AGIT*.

4.1.3 Uporaba modela *AGIT* v akademskem okolju

Posamezne dele modela smo najprej preizkusili v akademskem okolju.

V študijskem letu 2008/2009 smo vključili izvedbo projekta z uporabo metodologije *Scrum* v praktični del poučevanja pri predmetu Tehnologija programske opreme na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Predmet so poslušali študenti univerzitetnega programa Računalništvo in informatika, smer Programska oprema, v zadnjem (osmem) semestru študija. Študenti so bili organizirani v sedem *Razvojnih skupin*, pri vseh je bil *Skrbnik Metodologije Scrum* nosilec predmeta. *Razvojne skupine* so imele na izbiro dva projekta. Prvi projekt je obsegal »Informacijski sistem splošne bolnišnice«, pri čemer je vlogo *Produktnega vodje* imel predstavnik podjetja za razvoj programske opreme, ki je bil tudi študent magistrskega študija. Drugi projekt je bil »Orodje za vodenje projektov po metodologiji Scrum«, pri tem projektu je imel vlogo *Produktnega vodje* nosilec predmeta.

Pri spremljanju učinkovitosti razvoja smo uporabili tedanjo verzija modela *AGIT*, s poudarkom na *Padajočem diagramu napredka* ter kazalnikih Časovni indeks (SPI) in Stroškovni indeks (CPI). Pred izvedbo posamezne vaje oziroma *Dnevnega Scrum sestanka* smo opravili izračune kazalnikov učinkovitosti in jih potem na *Dnevnem Scrum sestanku* predstavili *Razvojni skupini,* ki je lahko komentirala odstopanja. Na *Retrospektivnem sestanku Iteracije* smo ocenili zadovoljstvo razvijalcev z uporabo vprašalnikov. Na zadnjem srečanju s študenti oziroma *Retrospektivnem sestanku Iteracije* smo predstavili rezultate vseh *Razvojnih skupin,* ki so potem lahko skupno komentirale potek razvoja in vpliv na končne rezultate. Rezultati so bolj podrobno opisani v [Mah09], [Mah10a], [Mah10b] in [Mah11a].

Glede na pozitivne izkušnje v študijskem letu 2008/2009, smo model uporabili tudi v študijskih letih, ki so sledila: 2009/2010, 2010/2011 in 2011/2012. V študijskem letu 2009/2010 smo prvič uporabili *Uporabniške zgodbe.* Večji poudarek smo dali sposobnosti ocenjevanja in planiranja ter zagotavljanju kakovosti in testiranju *Uporabniških zgodb.* Povzetek pridobljenih izkušenj je opisan v [Mah11b], [Mah12a]. Podatki, pridobljeni v tem študijskem letu so bili izhodišče za raziskavo točnosti ocen z uporabo metode *Načrtovalni poker*, kar je podrobneje opisano v [Mah12b].

4.1.4 Uporaba modela *AGIT* v gospodarskem okolju

Leta 2011 smo model *AGIT* preizkusili tudi v gospodarskem okolju [Mah12c], kar bomo v doktorski disertaciji podrobneje opisali kot študijo primera v petem poglavju, rezultate, ki se nanašajo na merjenje napredka, pa smo objavili v [Mah12d].

V nadaljevanju bo podan opis izpopolnjene različice modela *AGIT*, ki ga bo Laboratorij za tehnologijo programske opreme v prihodnosti še dopolnjeval in razvijal. Pri opisu trenutnega stanja modela *AGIT* smo kot izhodišče uporabili prvo različico modela iz leta 2007, nadgrajeno s spoznanji in izkušnjami, ki smo jih pridobili v obdobju od leta 2007 do danes.

4.2 Opis modela *AGIT*

4.2.1 Opis razvojnega procesa

Izvorni proces razvoja programske opreme z uporabo metodologije *Scrum* smo predstavili v drugem poglavju doktorske disertacije. V tem poglavju bomo opisali proces razvoja programske opreme z uporabo metodologije *Scrum*, kot ga uporabljamo v modelu *AGIT*.

4.2.2 Deležniki procesa

Izvorne vloge pri procesu razvoja programske opreme z uporabo metodologije *Scrum* smo predstavili v drugem poglavju doktorske disertacije. V modelu *AGIT* smo kot deležnike pri procesu razvoja programske opreme z uporabo metodologije *Scrum* opredelili naslednje vloge:

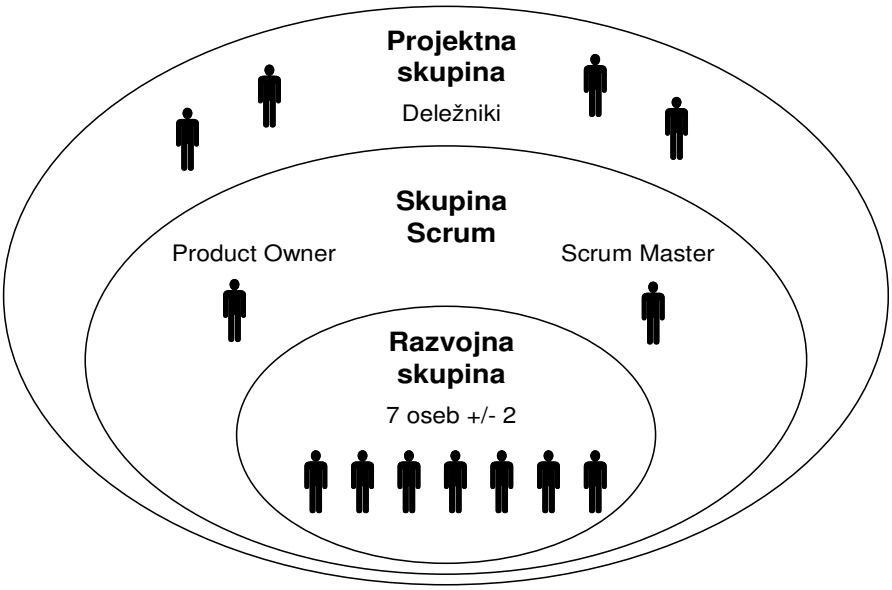
- Poslovodstvo.
- *Produktni vodja,*
- *Razvojna skupina* in
- *Skrbnik Metodologije Scrum.*

Deležnik Poslovodstvo ni del osnovnega nabora vlog pri metodologiji *Scrum*, čeprav je naveden v poznejših delih iz tega področja. Poslovodstvo odloča o proračunu in je ključnega pomena z vidika podpore uvedbi metodologije *Scrum*. Čeprav so lahko cilji vseh treh izvornih *Scrum* vlog izpolnjeni, to še ne pomeni, da bo poslovodstvo zadovoljno z rezultati, v kolikor ne bo upoštevan finančni vidik. Poslovodstvo financira *Razvojno skupino* in pričakuje, da se bodo vložena finančna sredstva s prodajo rezultatov razvoja povrnila in/ali prinesla dobiček. Zato ima v modelu *AGIT* deležnik Poslovodstvo zelo pomembno vlogo.

Kot lastnika procesa razvoja programske opreme z uporabo metodologije *Scrum* smo opredelili vlogo *Skrbnika Metodologije Scrum*, ki je odgovoren za odpravljanje *Ovir* ter skrbi za skladnost procesa s pravili metodologije *Scrum*. Glede na pomen finančnega vidika za spremljanje učinkovitosti, smo predpostavili, da ima *Skrbnik Metodologije Scrum* tudi nalogo zagotoviti, da proračun ne bo prekoračen.

Na sliki 17 so predstavljeni deležniki procesa razvoja programske opreme z uporabo metodologije *Scrum.*

Slika 17: Deležniki procesa razvoja programske opreme z uporabo metodologije Scrum



Vir: [Lay12]

Pri opredelitvi deležnikov procesa smo torej upoštevali vse tri *Scrum* vloge ter dodatno vlogo Poslovodstvo, pri čemer smo kot lastnika procesa opredelili vlogo *Skrbnik Metodologije Scrum* in tej vlogi dodelili dodatno zadolžitev spremljanja proračuna projekta. Vse štiri vloge so predstavljene v tabeli 19.

Tabela 19: Deležniki procesa razvoja programske opreme z uporabo metodologije Scrum

	Deležnik 1: Poslovodstvo	
Deležnik 2: Produktni vodja		Deležnik 3: Razvojna skupina
	Deležnik 4: Skrbnik *Metodologije Scrum*	

4.2.3 Cilji deležnikov procesa

V tem koraku smo za vsakega izmed deležnikov opredelili cilje, ki so značilni za posamezni vidik. Nekateri cilji so skupni več deležnikom, vendar bomo zaradi boljše preglednosti modela, pri vsakem deležniku navedli le najbolj pomembne cilje.

Cilj Poslovodstva je pravočasna informacija o učinkovitosti projekta, ki omogoča pravočasno ukrepanje ter izboljšanje kakovosti, ki vpliva predvsem na doseganje zadovoljstva strank. Z doseganjem cilja Poslovodstva je omogočeno pravočasno ukrepanje pri odstopanjih glede obsega, roka, stroškov in kakovosti. Pri tem bi izpostavili, da Poslovodstvo spremlja informacije o učinkovitosti projekta predvsem na ravni *Iteracije* ali *Izdaje*, za razliko od *Skrbnika Metodologije Scrum*, ki spremlja informacije o učinkovitosti projekta dnevno na *Dnevnem Scrum sestanku* in lahko *Ovire*, ki jih ne more rešiti sam, prepusti v reševanje Poslovodstvu.

Cilj *Produktnega vodje* je zadovoljstvo strank, ki ga dosežemo s kakovostnim izdelkom, ki je pravočasno na voljo. Z doseganjem cilja *Produktnega vodje* je zagotovljeno, da bo rezultat procesa izdelek, ki bo prinašal dodano vrednost.

Cilj *Razvojne skupine* je zadovoljstvo razvijalcev, ki ga dosežemo z ustreznim delovnim okoljem in enakomerno delovno obremenitvijo. Z doseganjem cilja *Razvojne skupine* je zagotovljena ustrezna motivacija za doseganje ciljev projekta.

Cilj *Skrbnika Metodologije Scrum* je nemoten potek projekta v skladu s pravili metodologije *Scrum*, kar je moč zagotoviti z učinkovitim odpravljanjem *Ovir.* Vloga *Skrbnika Metodologije Scrum* je z vidika zagotavljanja učinkovitost najbolj pomembna, ker je *Skrbnik Metodologije Scrum* edina vloga, ki lahko kadarkoli komunicira z vsemi deležniki. Z doseganjem cilja *Skrbnika Metodologije Scrum* je zagotovljena skladnost procesa z metodologijo *Scrum* in izkoriščanje prednosti te metodologije, kot so fleksibilnost in prilagojenost pogostim spremembam.

Čeprav smo navedli cilje za vsakega deležnika posebej, se cilji medsebojno ne izključujejo. Cilj *Produktnega vodje* je istočasno tudi cilj vseh ostalih deležnikov, saj vsi želijo ustvariti dodatno vrednost. *Razvojna skupina* prav tako želi biti seznanjena z informacijami o učinkovitosti, ki ji omogoča, da dnevno izboljšuje ocene preostalega časa in s tem vpliva na potrebe po nadurnem delu, ki so ponavadi odraz preveč optimističnih ocen in posledično »izrednih razmer« ob koncu *Iteracije*. *Skrbnik Metodologije Scrum* je sicer usmerjen v odpravljanje *Ovir*, vendar poleg *Ovir*, s katerimi ga seznani *Razvojna skupina*, lahko s pravočasno informacijo o učinkovitosti projekta ugotovi tudi *Ovire*, za katere ni nujno, da se jih *Razvojna skupina* zaveda. Npr. če *Skrbnik Metodologije Scrum* ugotovi, da se število preostalih ur povečuje, namesto, da bi se zmanjševalo, pri čemer *Razvojna skupina* ni poročala o *Ovirah*, lahko sproži diskusijo o razlogih za povečevanje preostalega časa ter njegove posledice na doseganje roka zaključka *Iteracije* in *Izdaje.* Iz navedenega je razvidno, da ima *Skrbnik Metodologije Scrum* ključno vlogo iz vidika projektnega vodenja, ker lahko ukrepa na dnevni ravni.

Skladno z opredelitvijo modela PPMS, doseganje vseh navedenih ciljev pomeni zadovoljstvo vseh deležnikov in doseganje učinkovitosti procesa. Cilji deležnikov so navedeni v Tabeli 20.

Tabela 20: Cilji deležnikov

	Deležnik 1: Poslovodstvo	
	Cilj: Pravočasna informacija o učinkovitosti projekta	
	Cilj: Izboljšanje kakovosti	
Deležnik 2: Produktni vodja		**Deležnik 3: *Razvojna skupina***
Cilj: Zadovoljstvo strank		*Cilj: Zadovoljstvo razvijalcev*
	Deležnik 4: Skrbnik *Metodologije Scrum*	
	Cilj: Nemoten potek projekta	
	Cilj: Učinkovito odpravljanje ovir	

4.2.4 Kazalniki procesa

Doseganje ciljev merimo s kazalniki, ki so lahko napovedovalni (ang. »leading«) ali pojasnjevalni (ang. »lagging«). Napovedovalni kazalniki se nanašajo na prihodnost in posredujejo informacijo o tem, ali cilj bo dosežen. Pojasnjevalni kazalniki se nanašajo na preteklost in posredujejo informacijo o tem, ali je cilj že dosežen.

Pri procesu razvoja programske opreme z uporabo metodologije *Scrum* se napovedovalni kazalniki nanašajo na proces, pojasnjevalni pa na rezultate procesa. Uporabili bomo kvantitativne in kvalitatitivne kazalnike.

V spodnji tabeli so navedeni vsi kazalniki. Za vsakega izmed kazalnikov je s črko »X« označeno, na kateri cilj se nanaša. Podobno kot pri določanju ciljev, tudi tukaj velja, da se nekateri kazalniki nanašajo na več ciljev.

Matrika kazalnikov in ciljev je opisana v Tabeli 21. Podroben opis kazalnikov je podan v naslednjem poglavju.

Tabela 21: Matrika kazalnikov in ciljev

ZŠ	Cilji → Kazalniki ↓	Zadovoljstvo strank	Zadovoljstvo razvijalcev	Nemoten potek projekta	Učinkovito odpravljanje ovir	Pravočasna informacija o učinkovitosti projekta	Izboljšanje kakovosti
K01	Doseganje obsega	X				X	
K02	Časovni indeks					X	
K03	Stroškovni indeks					X	
K04	Učinkovitost dela			X		X	
K05	Sprememba *Hitrosti*			X		X	
K06	Napredek *Izdaje*			X		X	
K07	Napredek *Iteracije*			X		X	
K08	Pogostost napak	X					X
K09	Stroški ponovnega dela						X
K10	Kvalitativna ocena zadovoljstva strank	X					
K11	Obseg nadurnega dela		X				
K12	Obseg sočasnega dela na več projektih		X				
K13	Kvalitativna ocena delovnih pogojev		X				
K14	BRE			X			
K15	Povprečno število *Ovir*				X		
K16	Povprečen čas za reševanje Ovir				X		

Struktura modela je predstavljena na Sliki 18, kjer so za vsakega deležnika navedeni cilji in kazalniki.

Slika 18: Struktura modela AGIT

	Deležnik 1: Poslovodstvo	
	Cilj: Pravočasna informacija o učinkovitosti projekta	
	K01-Doseganje obsega	
	K02-Časovni indeks	
	K03-Stroškovni indeks	
	K04-Učinkovitost dela	
	K05-Sprememba *Hitrosti*	
	K06-Napredek *Izdaje*	
	K07-Napredek *Iteracije*	
	Cilj: Izboljšanje kakovosti	
	K08-Pogostost napak	
	K09-Stroški ponovnega dela	
Deležnik 2: Produktni vodja		**Deležnik 4: Razvojna skupina**
Cilj: Zadovoljstvo strank		*Cilj: Zadovoljstvo razvijalcev*
K01-Doseganje obsega		K11-Obseg nadurnega dela
K08-Pogostost napak		K12- Obseg sočasnega dela na več projektih
K10-Kvalitativna ocena zadovoljstva stranik		K13-Kvalitativna ocena delovnih pogojev
	Deležnik 3: Skrbnik *Metodologije Scrum*	
	Cilj: Nemoten potek projekta	
	K05-Sprememba *Hitrosti*	
	K06-Napredek *Izdaje*	
	K07-Napredek *Iteracije*	
	K14-BRE	
	K04-Učinkovitost dela	
	Cilj: Učinkovito odpravljanje ovir	
	K15-Povprečno število nerešenih ovir	
	K16-Povprečen čas za reševanje ovir	

4.3 Podroben opis kazalnikov

4.3.1 Kazalniki za vlogo Poslovodstvo

4.3.1.1 Uvod

V tem poglavju bomo opisali kazalnike za vlogo Poslovodstvo, ki so navedeni v Tabeli 22.

Tabela 22: Kazalniki za vlogo Poslovodstvo

Deležnik 1: Poslovodstvo
Cilj: Pravočasna informacija o učinkovitosti projekta
K01-Doseganje obsega
K02-Časovni indeks
K03-Stroškovni indeks
K04-Učinkovitost dela
K05-Sprememba *Hitrosti*
K06-Napredek *Izdaje*
K07-Napredek *Iteracije*
Cilj: Izboljšanje kakovosti
K08-Pogostost napak
K09-Stroški ponovnega dela

4.3.1.2 Doseganje obsega (K01)

Doseganje obsega kaže doseganje vsebinskih ciljev in je enako razmerju med številom potrjenih *Uporabniških zgodb* in številom vseh *Uporabniških zgodb* na ravni *Iteracije* ali na ravni *Izdaje.*

Želena vrednost kazalnika je ena. Če je vrednost kazalnika enaka ena, pomeni, da so uresničene vse *Uporabniške zgodbe* v *Izdaji* ali v *Iteraciji.* Če je vrednost kazalnika večja od ena, pomeni, da je potrjenih več *Uporabniških zgodb*, kot je bilo načrtovano. Če pa je vrednost kazalnika manjša od ena, pomeni, da niso potrjene vse načrtovane *Uporabniške zgodbe*. V tem primeru je potrebno ugotoviti vpliv nepotrjenih *Uporabniških zgodb* na naslednjo *Izdajo* ali *Iteracijo.*

Skrbnik Metodologije Scrum spremlja doseganje obsega in oceni sposobnost *Razvojne skupine*, da doseže zastavljene cilje. Pri tem je pomembna opredelitev kriterija za zaključek *Uporabniške zgodbe* (koncept »done«).

4.3.1.3 Agilna prislužena vrednost

Model *AGIT* uvaja računanje časovnega indeksa (SPI) in stroškovnega indeksa (CPI) na dnevni ravni. Izračun agilne prislužene vrednosti na dnevni ravni je podan v [Mah07a]. Predlagane formule omogočajo izračun SPI in CPI po *Dnevnem Scrum sestanku*. Predpostavka izračuna je, da je obseg dela, ki bi moral biti dokončan v določenem trenutku, proporcionalen času, ki je pretekel od začetka *Iteracije.*

Komponente izračuna SPI and CPI na dnevni ravni so predstavljene v Tabeli 23.

Tabela 23: Ključne komponenta izračuna SPI and CPI na dnevni ravni

Oznaka	Opis v slovenščini	Opis v angleščini
j	Naloga j	Task j
d	Dan d v Sprintu	Day d of a Sprint
CEH	Cena inženirske ure	Cost of engineering hour
DE	Število pretečenih dni od začetka Sprinta	Days elapsed in sprint
SL	Dolžina Sprinta	Sprint length
ER	Pravilo za prisluženo vrednost	Earning rule
WS	Porabljen čas	Work spent
WR	Preostalo delo	Work remaining
WR_{init}	Preostalo delo - začetna ocena	Work remaining - initial estimate

Pravilo za prisluženo vrednost (ER) se izračuna za vsako nalogo *j* v *Seznamu Nalog Iteracije* na dan *d* v *Iteraciji* in predstavlja oceno za delež *Naloge*, ki je že končana.

ER je enak razmerju med seštevkom dejansko porabljenih ur in seštevkom dejansko porabljenih ur ter trenutne ocene preostalega dela, ki je potrebno za zaključek *Naloge*. Izračunamo ga s pomočjo formule (6), pomen oznak je podan v Tabeli 23.

$$ER_j = \frac{\sum_{i=1}^{d-1} WS_{i,j}}{\sum_{i=1}^{d-1} WS_{i,j} + WR_{d,j}} \tag{6}$$

Pravilo za prisluženo vrednost je primerljivo z odstotkom opravljenega dela. Na začetku *Iteracije* je *ER* enako 0 za vse naloge, ob koncu *Iteracije* pa bi moral biti enak 1 za vse *Naloge.* V vmesnem času je *ER* enako 1 samo za tiste *Naloge*, ki so potrjene.

4.3.1.4 Časovni indeks (K02)

Časovni indeks ali SPI je razmerje med prisluženo vrednostjo in načrtovano vrednostjo. Prislužena vrednost je enaka vsoti produktov pravila za prisluženo vrednost in začetne ocene preostalega časa za izdelavo *Naloge*. Načrtovana vrednost je enaka seštevku začetnih ocen preostalega časa za vse *Naloge* in razmerja med preteklim in skupnim številom delovnih dni v *Iteraciji.* Pri izračunu SPI je pomembna točnost začetne ocene preostalega dela za posamezno *Nalogo*, ki je osnova za izračun referenčnega stanja.

SPI izračunamo s pomočjo formule (7).

$$SPI = \frac{\sum_{j=1}^{n} ER_j \cdot WR_{init,j}}{\sum_{j=1}^{n} WR_{init,j} * \frac{DE}{SL}} \cdot \tag{7}$$

Podobno kot pri *Padajočem diagramu napredka Iteracije*, analiza časovnega indeksa (SPI) omogoča *Skrbniku Metodologije Scrum*, da pravočasno ugotovi tveganje zamude.

Želena vrednost kazalnika SPI je večja ali enaka ena. Če je vrednost SPI enaka 1, pomeni, da *Razvojna skupina* lahko doseže rok. Če je vrednost večja od 1, pomeni, da *Razvojna skupina* lahko konča delo pred rokom. Če je vrednost manjša od 1, pomeni, da *Razvojna skupina*

zaostaja in da roka ne bo dosegla. Naloga *Skrbnika Metodologije Scrum* je, da ugotovi razloge za zaostajanje in ukrepa.

4.3.1.5 Stroškovni indeks (K03)

Stroškovni indeks ali CPI je razmerje med prisluženo vrednostjo in dejanskimi stroški. Prislužena vrednost je enaka vsoti produktov pravila za prisluženo vrednost, začetne ocene preostalega časa za izdelavo naloge in cene inženirske ure. Načrtovana vrednost je enaka seštevku produktov dejansko porabljenih ur in cene inženirske ure.

Stroškovni indeks je edini indikator, ki vsebuje podatke o stroških in omogoča *Skrbniku Metodologije Scrum*, da spremlja porabo projektnega proračuna. V modelu *AGIT* so stroški odvisni od porabljenega časa. Upoštevajoč kazalnik CPI lahko *Skrbnik Metodologije Scrum* pravočasno ugotovi, da bo proračun prekoračen in pravočasno ukrepa.

CPI izračunamo s pomočjo formule (8).

$$CPI = \frac{\sum_{j=1}^{n} ER_j \cdot WR_{init,j} \cdot CEH_j}{\sum_{i=1}^{DE} \sum_{j=1}^{n} WS_{i,j} \cdot CEH_j} \tag{8}$$

Želena vrednost kazalnika CPI je večja ali enako ena. Če je vrednost CPI enaka 1 pomeni, da so dejanski stroški razvoja enaki načrtovanim. Če je vrednost večja od 1 pomeni, da bo *Razvojna skupina* lahko dosegla prihranek, ker ne bo porabila celotnega proračuna. Če je vrednost manjša od 1 pomeni, da je *Razvojna skupina* porabila več, kot je bilo načrtovano in da bo verjetno prekoračila proračun do konca *Iteracije.* Naloga *Skrbnika Metodologije Scrum* je, da ugotovi razloge za prekoračitev in ukrepa.

4.3.1.6 Učinkovitost dela (K04)

Kazalnik učinkovitost dela kaže, kako je vloženo delo vplivalo na zmanjšanje preostalega časa potrebnega za potrjevanje vseh *Nalog* v *Iteraciji.*

Učinkovitost dela (oznaka *WE*) med dnevi *d1* in *d2* izračunamo s pomočjo formule (9), v kateri je pomen oznak naslednji:

- *WR* pomeni preostali čas,
- *WS* pomeni porabljeni čas,
- *n* pomeni število *Nalog* v *Seznamu Nalog Iteracije*.

$$WE(d1, d2) = \frac{\sum_{j=1}^{n} WR_{d1,j} - \sum_{j=1}^{n} WR_{d2,j}}{\sum_{i=d1}^{d2-1} \sum_{j=1}^{n} WS_{i,j}} \tag{9}$$

Želena vrednost kazalnika je najmanj 1. Če je vrednost enaka 1 pomeni, da je *Razvojna skupina* točno ocenila preostali čas. Če je vrednost kazalnika večja od 1 pomeni, da je

Razvojna skupina vložila manj časa, kot je prvotno načrtovala oziroma, da je dejansko porabljen čas bil manjši od ocenjenega časa (pesimistična ocena). Če je vrednost kazalnika manjša od 1 pomeni, da je *Razvojna skupina* vložila več časa, kot je prvotno načrtovala oziroma, da je dejansko porabljen čas bil večji od ocenjenega časa (optimistična ocena). Razlogi so lahko npr. v večji zahtevnosti *Nalog*, spremembi sestave *Razvojne skupine*. Naloga *Skrbnika Metodologije Scrum* je, da v takem primeru ugotovi vzroke in odpravi *Ovire*, zaradi katerih je *Razvojna skupina* porabila več časa ali pa zagotovi, da *Razvojna skupina* v prihodnje poveča ocene preostalega časa, ter temu ustrezno prilagodi *Seznam Nalog Iteracije.* Kazalnik učinkovitost dela omogoča analizo razlik med ocenjenim in dejanskim časom ter izboljšanje ocen preostalega časa.

4.3.1.7 Sprememba *Hitrosti* (K05)

Hitrost je število točk *Uporabniških zgodb* iz *Seznama Uporabniških zgodb,* ki jih *Razvojna skupina* lahko realizira v eni *Iteraciji*.

Dejanska *Hitrost Razvojne skupine* je število realiziranih točk ob zaključku *Iteracije*. Načrtovana *Hitrost* t je ocena števila realiziranih točk v tekoči *Iteraciji*. Načrtovana *Hitrost* predstavlja osnovo za ocenjevanje na začetku *Iteracije*, ko se odloča o tem koliko *Uporabniških zgodb* bo uvrščeno v tekočo *Iteracijo*. Sprememba *Hitrosti* je razlika med dejansko in načrtovano *Hitrostjo*.

Analiza spremembe *Hitrosti* omogoča *Skrbniku Metodologije Scrum*, da oceni čas, ki je potreben za dokončanje *Uporabniških zgodb* iz *Seznama Uporabniških zgodb*, pod pogojem, da *Razvojna skupina* ohrani doseženo *Hitrost*. S spremljanjem spremembe *Hitrosti* lahko *Skrbnik Metodologije Scrum* pravočasno ugotovi, da je *Hitrost Razvojne skupine* premajhna in da bo potrebno zmanjšati *Seznam Uporabniških zgodb*.

Hitrost je specifična za posamezno *Razvojna skupina* in ni primerljiva med *Razvojnimi skupinami*.

Želena vrednost kazalnika je večja ali enaka nič. Če je vrednost kazalnika enaka nič, potem velja, da je *Hitrost* stabilna in bo zastavljeni rok *Izdaje* dosežen. Če je vrednost kazalnika večja od nič, pomeni, da je *Razvojna skupina* hitrejša kot je bilo načrtovano in da bo zaključila delo pred rokom oziroma bo lahko do roka razvijala programsko opremo za dodatne *Uporabniške zgodbe*. Če pa je vrednost kazalnika manjša od nič, potem to lahko pomeni *Oviro* za doseganje zastavljenega roka *Izdaje.* Naloga *Skrbnika Metodologije Scrum* je, da ugotovi vzrok znižanja *Hitrosti* in pomaga *Razvojni skupini*, da poveča *Hitrost*. Če povečanje *Hitrosti* ni možno, potem je potrebno pridobiti dovoljenje *Produktnega vodje* za izločitev nekaterih *Uporabniških zgod* iz *Seznama Uporabniških zgodb.*

4.3.1.8 Napredek *Izdaje* (K06)

Napredek *Izdaje* je razlika med dejanskim in načrtovanim številom nepotrjenih *Uporabniških zgodb,* ki jo izračunamo na koncu *Iteracije,* ko *Produktni vodja* potrdi *Uporabniške zgodbe*.

Dejansko število nepotrjenih *Uporabniških zgodb* na koncu *Iteracije* je enako številu *Uporabniških zgodb*, ki jih je potrebno narediti do konca *Izdaje* in še niso narejene. Načrtovano število nepotrjenih *Uporabniških zgodb* na koncu *Iteracije* je enako nič.

Padajoči diagram spremljanja napredka Izdaje prikazuje preostali obseg dela na začetku vsake *Iteracije,* izražen z vsoto točk nepotrjenih *Uporabniških zgodb* v *Seznamu Uporabniških zgodb*. *Padajoči diagram spremljanja napredka Izdaje* omogoča sprotno spremljanje dejanskega poteka celotnega projekta, glede na pričakovano stanje ter izračun datuma zaključka projekta ob trenutni *Hitrosti*. *Padajoči diagram spremljanja napredka Izdaje* omogoča *Skrbniku Metodologije Scrum* pregled nad uresničevanjem *Seznama Uporabniških zgodb*. Z uporabo ostalih kazalnikov lahko *Skrbnik Metodologije Scrum* oceni rok *Izdaje* ob nespremenjenih pogojih in potem predlaga spremembo *Seznama Uporabniških zgodb.*

Želena vrednost kazalnika je večja ali enaka nič. Če je vrednost kazalnika enaka nič, potem velja, da bo rok *Izdaje* dosežen. Če je vrednost kazalnika večja od nič, potem bo *Izdaja* potrjena pred rokom ali pa bo do roka potrjeno več *Uporabniških zgodb*, kot je bilo načrtovano. Če pa je vrednost kazalnika manjša od nič, potem to lahko pomeni *Oviro* za doseganje roka *Izdaje*. Naloga *Skrbnika Metodologije Scrum* je, da ugotovi vzrok odstopanja in pomaga *Razvojni skupini*, da doseže rok *Izdaje*. To lahko stori s spremembo *Seznama Uporabniških zgodb*, ki jo odobri *Produktni vodja*.

4.3.1.9 Napredek *Iteracije* (K07)

Napredek *Iteracije* je razlika med poročanim in načrtovanim preostalim časom v *Iteraciji,* ki jo izračunamo na *Dnevnem Scrum sestanku*, ko člani *Razvojne skupine* poročajo ocene preostalega časa do zaključka svojih *Nalog*.

Poročani preostali čas je enak seštevku preostalega časa za vse *Naloge* v *Seznamu Nalog Iteracije*. Načrtovan preostali čas je enak linearnemu zmanjšanju preostalega časa.

Padajoči diagram spremljanja napredka Iteracije prikazuje preostali obseg dela na začetku vsakega *Dnevnega Scrum sestanka,* izražen v vsoti ur potrebnih za potrjevanje *Nalog* v *Seznamu Nalog Iteracije*. Diagram omogoča sprotno spremljanje dejanskega poteka tekoče *Iteracije,* glede na pričakovano stanje in izračun datuma zaključka *Iteracije* ob trenutni *Hitrosti*. *Padajoči diagram spremljanja napredka Iteracije* omogoča *Skrbniku Metodologije Scrum* spremljanje preostalega časa, kot ga je ocenila *Razvojna skupina*. Z uporabo ostalih kazalnikov (hitrost, pravilnost ocen) lahko *Skrbnik Metodologije Scrum* oceni čas dokončanja *Iteracije* ob nespremenjenih pogojih.

Želena vrednost kazalnika je večja ali enaka nič. Če je vrednost kazalnika enaka nič, potem velja, da bo rok *Iteracije* dosežen. Če je vrednost kazalnika večja od nič, potem bo *Iteracija* zaključena pred rokom ali pa bo do roka potrjeno več *Uporabniških zgodb,* kot je bilo načrtovano. Če pa je vrednost kazalnika manjša od nič, potem to lahko pomeni *Oviro* za doseganje roka *Iteracije.* Naloga *Skrbnika Metodologije Scrum* je, da ugotovi vzrok odstopanja in pomaga *Razvojni skupini*, da realizira vse *Naloge* do konca *Iteracije*.

4.3.1.10 Pogostost napak (K08)

Pogostost napak (ang. Defect Density) je število napak na tisoč vrstic kode ali KLOC (kilo lines of code) [Fen99]. Izračunamo ga za posamezno *Uporabniško zgodbo* ter seštejemo na ravni *Iteracije* ali *Izdaje.* Na ravni *Iteracije* upoštevamo napake, ki so ugotovljene na *Potrjevalnem sestanku Iteracije*. Na ravni *Izdaje* upoštevamo napake, ki so ugotovljene v določenem obdobju po *Izdaji.*

Želena vrednost kazalnika je nič. Če je število napak enako nič, potem je cilj dosežen. Če je število napak večje od nič pomeni, da so v kodi napake. Naloga *Skrbnika Metodologije Scrum* je, da ugotovi vzroke napak in jih odpravi. Možni vzroki napak so: časovni pritisk, zaradi katerega je skrajšan čas testiranja, pomanjkljiva opredelitev koncepta "done", ki ne vključuje celovitega testiranja in podobno.

4.3.1.11 Stroški ponovnega dela (K09)

Stroški ponovnega dela (ang. Costs of Rework) so produkt števila ur, porabljenih za ponovno delo in cene inženirske ure. Ponovno delo se nanaša na dodaten čas, ki je porabljen za realizacijo popravkov, ki so potrebni za dokončanje *Naloge*. Razlog za dodaten čas je lahko sprememba *Uporabniške zgodbe,* odpravljanje napak in podobno. Za namen zbiranja podatkov o dodatnem delu je potrebno opredeliti kategorijo *Naloge* "ponovno delo".

Želena vrednost kazalnika je nič. Če je vrednost kazalnika enaka nič, potem je cilj dosežen. Če je vrednost kazalnika večja od nič, pomeni, da je bil del proračuna projekta porabljen za neplanirano delo, ki ni del *Seznama Nalog Iteracije*. V takih primerih je *Skrbnik Metodologije Scrum* dolžan ugotoviti vzroke ponovnega dela in jih odpraviti. Možni razlogi so: pomanjkljivo opredeljene *Uporabniške zgodbe*, zaradi katerih je bilo potrebno že potrjene *Uporabniške zgodbe* popravljati, pomanjkljiva kakovost programov, zaradi katerih je bilo potrebno vložiti dodaten čas za odpravljanje napak in podobno.

4.3.2 Kazalniki za vlogo *Produktni vodja*

4.3.2.1 Uvod

V tem poglavju bomo opisali le tiste kazalnike za vlogo *Produktni vodja*, ki so navedeni v Tabeli 24 in jih še nismo opisali v predhodnih poglavjih.

Tabela 24: Kazalniki za vlogo Produktni vodja

Deležnik 2: Produktni vodja
Cilj: Zadovoljstvo strank
K01-Doseganje obsega
K08-Pogostost napak
K10-Kvalitativna ocena zadovoljstva strank

4.3.2.2 Kvalitativna ocena zadovoljstva strank (K10)

Kvalitativna ocena zadovoljstva strank se pridobi iz rezultatov ankete o zadovoljstvu strank na *Potrjevalnem sestanku Iteracije* (ocene so od 1 do 5, pri čemer je 1 najslabša ocena in 5 najboljša ocena). Vprašanja v anketi se lahko nanašajo na kakovost izdelka, ustreznost cene, fleksibilnost pri upravljanju sprememb zahtev, kakovost sodelovanja z *Razvojno skupino*, kakovost usposabljanja in dokumentacije.

Želena vrednost kazalnika je 3 ali več. Če je vrednosti kazalnika 3 ali več, potem so stranke zadovoljne in je cilj dosežen. Če pa je vrednost kazalnika manjša od 3, potem to lahko pomeni *Oviro.* V tem primeru mora *Skrbnik Metodologije Scrum* ugotoviti vzroke in oceniti vpliv na potek tekoče *Iteracije* ter *Izdaje*. Možni vzroki nizke ocene zadovoljstva strank so lahko pomanjkljivo sodelovanje z *Produktnim vodjo* in posledično neustrezne *Uporabniške zgodbe*, zamude pri razvoju, nepotrjene *Uporabniške zgodbe*, napake v delovanju potrjenih *Uporabniških zgodb* in podobno. V Tabeli 25 je podan primer vprašalnika.

Tabela 25: Primer vprašalnika

1	Sodelovanje z *Razvojno skupino* v času razvoja
2	Odzivnost *Razvojne skupine* na želje, spremembe *Uporabniških zgodb*
3	Popolnost izdelane rešitve (Ali so bile realizirane vse *Uporabniške zgodbe*, dogovorjene na *Načrtovalnem sestanku Iteracije*?)
4	Skladnost izdelane rešitve s pričakovanji (Ali se izdelana programska oprema ujema z vašimi pričakovanji na začetku *Iteracije*?)
5	Intuitivnost Uporabniškega vmesnika (Ali je sistem enostaven za uporabo brez obsežnega predhodnega izobraževanja?)
6	Uporabnost/Koristnost izdelanih programov (Ali izdelani programi olajšajo delo uporabniku?)
7	Odsotnost napak, zanesljivost delovanja, robustnost (Ali so izdelani programi brez napak? Ali izdelana rešitev deluje stabilno, brez prekinitev?)
8	Kakovost in pravočasnost oddanih *Uporabniških zgodb*
9	Splošna ocena zadovoljstva z delom *Razvojne skupine*
10	Splošna ocena kakovosti izdelanih programov

4.3.3 Kazalniki za vlogo *Razvojna skupina*

4.3.3.1 Uvod

V tem poglavju bomo opisali kazalnike za vlogo *Razvojna skupina*, ki so navedeni v Tabeli 26.

Tabela 26: Kazalniki za vlogo Razvojna skupina

Deležnik 4: Razvojna skupina
Cilj: Zadovoljstvo razvijalcev
K11-Obseg nadurnega dela
K12-Obseg sočasnega dela na več projektih
K13-Kvalitativna ocena delovnih pogojev

4.3.3.2 Obseg nadurnega dela (K11)

Obseg nadurnega dela je razmerje med seštevkom dejansko opravljenih ur in seštevkom ur, ki so bile na začetku *Iteracije* predvidene za posameznega razvijalca, upoštevajoč njegovo delo na drugih projektih, praznike, dopust, bolniško odsotnost in podobno. Običajno je delavnik dolg osem ur. Če je razvijalec delal več kot osem ur, potem smo to upoštevali kot nadurno delo.

Število pričakovanih delovnih ur izračunamo z uporabo formule (10), pri čemer imajo oznake naslednji pomen: *m* pomeni član *Razvojne skupine* (ang. member),

- *SL* pomeni dolžina *Iteracije* (ang. Sprint Length),
- *AD* pomeni število administrativnih dni ali dni, ko član *Razvojne skupine* ne dela (npr. praznik, odsotnost zaradi bolezni),
- *PTME* pomeni odstotek udeležbe člana *Razvojne skupine* pri projektu (ang. Percentage of Team Member's Engagement in the Project),
- *OT* pomeni obseg nadurnega dela (ang. Overtime),
- *t* pomeni *Razvojna skupina*,
- *WS* pomeni porabljen čas,
- *NTM* pomeni število članov *Razvojne skupine* (ang. Number of Team Members).

$$EH_m = 8 \cdot (SL - AD) \cdot PTME_m / 100 \tag{10}$$

Obseg nadurnega dela je izračunan z uporabo formule (11), pri čemer imajo oznake naslednji pomen: *m* pomeni član *Razvojne skupine* (ang. member),

- *OT* pomeni obseg nadurnega dela (ang. Overtime),
- *t* pomeni *Razvojna skupina*,
- *WS* pomeni porabljen čas,
- *NTM* pomeni število članov *Razvojne skupine* (ang. Number of Team Members),
- *EH* pomeni število pričakovanih delovnih ur (ang. Expected Hours) za posameznega člana *Razvojne skupine*.

$$OT_t = \frac{\sum_{i=1}^{SL} \sum_{j=1}^{n} WS_{i,j}}{\sum_{m=1}^{NTM_t} EH_m} \tag{11}$$

Želena vrednost obsega nadurnega dela je 1 ali manj. Če je vrednost kazalnika enaka ali manj od 1, potem ni bilo nadurnega dela. Če pa je vrednost kazalnika večja od 1, potem je bilo izvajano nadurno delo in je tim delal več ur, kot je bilo predvideno. V tem primeru mora *Skrbnik Metodologije Scrum* ugotoviti vzroke nadurnega dela in jih upoštevati pri načrtovanju naslednje *Iteracije* ali *Izdaje*. Možni vzroki so časovni pritisk, sprememba sestave tima, neenakomerna obremenitev, nepredvidena odsotnost in podobno.

4.3.3.3 Obseg sočasnega dela na več projektih (K12)

Obseg sočasnega dela na več projektih je razmerje med seštevkom velikosti vseh *Razvojnih moštev* in skupnim številom razvijalcev.

Kazalnik izračunamo z uporabo formule (12), pri čemer imajo oznake naslednji pomen:

- *CP* pomeni obseg sočasnega dela na več projektih (ang. Concurrent Projects),
- *NTM* pomeni število članov posameznega *Razvojne skupine* (ang. Number of Team Members),
- *ND* pa pomeni skupno število razvijalcev (ang. Number of Developers) za vse projekte.

$$CP = \frac{\sum_t NTM_t}{ND} \tag{12}$$

Želena vrednost kazalnika je 1. Če je vrednost enaka 1 pomeni, da ni članov *Razvojne skupine*, ki bi delali na vzporednih projektih oziroma bi istočasno delali na več projektih. Če pa je vrednost večja od 1 pomeni, da so nekateri člani *Razvojne skupine* istočasno zasedeni tudi z delom na drugih projektih, kar lahko pomeni *Oviro*. Možni razlogi so, da imajo člani tima posebna znanja ali izkušnje, ki so potrebna na večjem številu projektov ali pa nadomeščajo člane tima v drugih projektih, ki so nepredvideno odsotni (npr. zaradi bolezni). V takšnih primerih mora *Skrbnik Metodologije Scrum* pozorno spremljati soodvisnost z ostalimi projekti, da lahko pravočasno ukrepa.

4.3.3.4 Kvalitativna ocena delovnih pogojev (K13)

Kvalitativna ocena delovnih pogojev se pridobi iz rezultatov ankete o delovnih pogojih na *Retrospektivnem sestanku Iteracije* (ocene so od 1 do 5, pri čemer je 1 najslabša ocena in 5 najboljša ocena). Vprašanja v anketi se lahko nanašajo na kakovost izdelka, ustreznost cene, fleksibilnost pri upravljanju sprememb zahtev, kakovost sodelovanja z *Razvojno skupino*, kakovost usposabljanja in dokumentacije.
Primer vprašalnika je predstavljen v [Mah09, Mah10a] in vsebuje vprašanja glede *Seznama Uporabniških zgodb*, *Seznama Nalog Iteracije*, tehničnih in vsebinskih težav, sodelovanja z ostalimi udeleženci v projektu ter splošna vprašanja.

Želena vrednost kazalnika je 3 ali več. Če je vrednosti kazalnika 3 ali več, potem so razvijalci zadovoljni in je cilj dosežen. Če pa je vrednost kazalnika manjša od 3, potem lahko pomeni *Oviro.* V tem primeru mora *Skrbnik Metodologije Scrum* ugotoviti vzroke in oceniti vpliv na potek tekoče *Iteracije* ter *Izdaje.* Možni vzroki nizke ocene zadovoljstva razvijalcev so lahko pomanjkljivo sodelovanje z *Produktnim vodjo*, *Skrbnikom Metodologije Scrum*, dinamika *Razvojne skupine*, obseg administrativnega dela, neenakomerna porazdelitev dela, obseg nadurnega dela, možnosti osebnega razvoja in podobno.

4.3.4 Kazalniki za vlogo *Skrbnik Metodologije Scrum*

4.3.4.1 Uvod

V tem poglavju bomo opisali le tiste kazalnike za vlogo *Skrbnik Metodologije Scrum*, ki so navedeni v Tabeli 27 in jih še nismo opisali v predhodnih poglavjih.

Tabela 27: Kazalniki za vlogo Skrbnik Metodologije Scrum

Deležnik 3: Skrbnik Metodologije Scrum
Cilj: Nemoten potek projekta
K05-Sprememba *Hitrosti*
K06-Napredek *Izdaje*
K07-Napredek *Iteracije*
K14-BRE (točke zgodb)
K04-Učinkovitost dela
Cilj: Učinkovito odpravljanje ovir
K15-Povprečno število nerešenih ovir
K16-Povprečen čas za reševanje ovir

4.3.4.2 BRE (K14)

BRE se nanaša na razliko med začetno oceno zahtevnosti *Uporabniških zgodb* in dejansko porabljenim časom za razvoj *Uporabniških zgodb*.

Analiza točnosti ocen omogoča *Skrbniku Metodologije Scrum,* da ugotovi stopnjo optimizma ocen posameznih razvijalcev in prispeva k čim bolj realnim ocenam preostalega časa.

Za oceno je bila uporabljena mera BRE, uravnoteženo merilo relativne napake (ang. balanced measure of relative error), ki jo izračunamo z uporabo formule (13), pomen oznak je naslednji:

- *A* (ang. actual) pomeni dejanska zahtevnost *Uporabniške zgodbe*,
- *E* (ang. estimate) pomeni ocenjena zahtevnost *Uporabniške zgodbe.*

$$BRE = \frac{|A - E|}{\min(A, E)} \tag{13}$$

Želena vrednost kazalnika *BRE* je 0. Če je vrednost kazalnika manjša od 0, pomeni, da je bila ocena pesimistična. Če je vrednost kazalnika večja od 0, pomeni, da je bila ocena optimistična. Naloga *Skrbnika Metodologije Scrum* je, da analizira točnost ocen in zagotovi, da tim postopoma izboljšuje ocene upoštevajoč odmike pri ocenjevanju v preteklem obdobju.

4.3.3.3 Povprečno število nerešenih *Ovir* (K15)

Povprečno število nerešenih *Ovir* je razmerje med številom nerešenih *Ovir* ob zaključku *Iteracije* in številom *Iteracij.*

Želena vrednost kazalnika je nič. Če je vrednost kazalnika enako nič pomeni, da se ovire sproti rešujejo in *Razvojna skupina* lahko svoj čas v celoti nameni razvoju. Če je vrednost

kazalnika večja od nič, potem je potrebno pri načrtovanju upoštevati čas za odpravljanje ovir. Naloga *Skrbnika Metodologije Scrum* je, da ovire rešuje na način, ki bo dolgoročno prispeval k njihovem zmanjševanju.

4.3.3.4 Povprečen čas za reševanje *Ovir* (K16)

Povprečno število nerešenih ovir ne pove veliko o zahtevnosti ovir, zato spremljamo tudi povprečen čas za reševanje ovir. Povprečen čas za reševanje *Ovir* je razmerje med časom, porabljenim za *reševanje Ovir* in številom *Ovir.*

Želena vrednost kazalnika je nič. Če je vrednost kazalnika enako nič pomeni, da se *Ovire* sproti rešujejo in *Razvojna skupina* lahko svoj čas v celoti nameni razvoju. Če je vrednost kazalnika večja od nič, potem je potrebno pri načrtovanju upoštevati čas za odpravljanje *Ovir. Naloga Skrbnika Metodologije Scrum je, da Ovire* rešuje na način, ki bo dolgoročno prispeval k njihovem zmanjševanju.

4.4 Zbiranje in prikazovanje podatkov

4.4.1 Ciljne in mejne vrednosti kazalnikov

Naslednji korak je opredelitev ciljnih in mejnih vrednosti kazalnikov. Ciljne vrednosti določajo idealno vrednost kazalnika, pri kateri ni potrebno ukrepanje. Mejne vrednosti kazalnikov določajo mejo, pri kateri je potrebno ukrepati. Mejne vrednosti so odvisne od odnosa poslovodstva do tveganja in so specifične za posamezni projekt. Mejne vrednosti imenujemo tudi sprožilci (ang. trigger), ker sprožijo opozorila ali alarme.

V primerih, ko pri razvoju programske opreme sodelujejo zunanji izvajalci, so mejne vrednosti povezane z določili pogodbe glede bonusov v primeru doseganja pogodbenih ciljev in malusov v primeru nedoseganja (npr. zamude, nepotrjene *Uporabniške zgodbe*, prekoračitev proračuna). Pri določanju mejnih vrednosti je pomembno, da se alarm sproži pravočasno. Metodologija *Scrum* omogoča dnevno spremljanje preostalega časa in statusov nalog ter posredno tudi statusov *Uporabniških zgodb.* Naloga *Skrbnika Metodologije Scrum* je, da analizira vpliv sprememb in odstopanj na končne rezultate. Zato je pomembno, da poleg kazalnikov na ravni *Naloge*, *Dnevnega Scrum sestanka* in *Iteracije*, opazuje tudi kazalnike na ravni *Izdaje*.

Nabor kazalnikov in ciljnih vrednosti za merjenje učinkovitosti razvoja programske opreme z uporabo metodologije *Scrum* je podan v Tabeli 28.

Tabela 28: Seznam kazalnikov in ciljnih vrednosti

ZŠ	Kazalnik	Sprožilec Minimalna vrednost	Ciljna vrednost	Sprožilec Maksimalna vrednost
K01	Doseganje obsega	*m*	1	*M*
K02	Časovni indeks	*m*	1	*M*
K03	Stroškovni indeks	*m*	1	*M*
K04	Učinkovitost dela	*m*	1	*M*
K05	Sprememba *Hitrosti*	*m*	0	*M*
K06	Napredek *Izdaje*	*m*	0	*M*
K07	Napredek *Iteracije*	*m*	0	*M*
K08	Pogostost napak	-	1	*M*
K09	Stroški ponovnega dela	*m*	0	*M*
K10	Kvalitativna ocena zadovoljstva strank (1-5)	*m*	*c*	*M*
K11	Obseg nadurnega dela	-	1	*M*
K12	Obseg sočasnega dela na več projektih	-	1	*M*
K13	Kvalitativna ocena delovnih pogojev (1-5)	*m*	*c*	*M*
K14	BRE (točke zgodb)	*m*	0	*M*
K15	Povprečno število nerešenih ovir	-	0	*M*
K16	Povprečen čas za reševanje ovir	*m*	*c*	*M*

Dopolnjen prikaz modela *AGIT* je podan na Sliki 19, pri čemer je *m* minimalna vrednost kazalnika, pri kateri se sproži opozorilo, *M* je maksimalna vrednost kazalnika, pri kateri se sproži opozorilo, *c* pa je ciljna vrednost kazalnika, ki jo določi poslovodstvo oziroma je določena v pogodbi.

Slika 19: Model AGIT

Deležnik 1: Poslovodstvo			
Cilj: Pravočasna informacija o učinkovitosti projekta			
K01-Doseganje obsega	*m*	1	*M*
K02-Časovni indeks	*m*	1	*M*
K03-Stroškovni indeks	*m*	1	*M*
K04-Učinkovitost dela	*m*	1	*M*
K05-Sprememba *Hitrosti*	*m*	0	*M*
K06-Napredek *Izdaje*	*m*	0	*M*
K07-Napredek *Iteracije*	*m*	0	*M*
Cilj: Izboljšanje kakovosti			
K08-Pogostost napak	-	1	*M*
K09-Stroški ponovnega dela	*m*	0	*M*

Deležnik 2: Produktni vodja			
Cilj: Zadovoljstvo strank			
K01-Doseganje obsega	*m*	1	*M*
K08-Pogostost napak	-	0	*M*
K10-Kvalitativna ocena zadovoljstva stranik	*m*	*c*	*M*

Deležnik 3: Razvojna skupina			
Cilj: Zadovoljstvo razvijalcev			
K11-Obseg nadurnega dela	-	1	*M*
K12- Obseg sočasnega dela na več projektih	-	1	*M*
K13-Kvalitativna ocena delovnih pogojev	*m*	*c*	*M*

Deležnik 4: Skrbnik *Metodologije Scrum*			
Cilj: Nemoten potek projekta			
K05-Sprememba *Hitrosti*	*m*	0	*M*
K06-Napredek *Izdaje*	*m*	0	*M*
K07-Napredek *Iteracije*	*m*	0	*M*
K14-BRE (točke zgodb)	*m*	0	*M*
K04-Učinkovitost dela	*m*	1	*M*
Cilj: Učinkovito odpravljanje ovir			
K15-Povprečno število nerešenih ovir	-	0	*M*
K16-Povprečen čas za reševanje ovir	*m*	*c*	*M*

Grafični prikaz trenutnih, ciljnih in mejnih vrednosti omogoča *Skrbniku Metodologije Scrum* in tudi drugim deležnikom, da transparentno spremljajo potek projekta in ocenijo verjetnost, da bo projekt dosegel cilje v zastavljenem obsegu, roku in v okviru proračuna. Več o prikazu podatkov je zapisano v nadaljevanju.

4.4.2 Zbiranje podatkov

Kazalnike procesa izračunamo s pomočjo osnovnih metrik. Za vsakega izmed ciljev so opredeljene metrike, podane v Tabeli 29.

Tabela 29: Seznam kazalnikov in osnovnih metrik

Oznaka	Kazalnik	Oznaka	Osnovna metrika
K01	Doseganje obsega	M01	Skupno število predvidenih *Uporabniških zgodb* v *Iteraciji*
		M02	Skupno število potrjenaih *Uporabniških zgodb* v *Iteraciji*
		M03	Skupno število predvidenih *Uporabniških zgodb* v *Izdaji*
		M04	Skupno število potrjenaih *Uporabniških zgodb* v *Izdaji*
K02	Časovni indeks	M05	WR (ang. Work Remaining) Preostali čas za posamezno *Nalogo* iz *Seznama Nalog Iteracije*
		M06	WS (ang. Work Spent) Porabljeni čas za posamezno *Nalogo* iz *Seznama Nalog Iteracije*
K03	Stroškovni indeks	M05	WR (ang. Work Remaining) Preostali čas za posamezno *Nalogo* iz *Seznama Nalog Iteracije*
		M06	WS (ang. Work Spent) Porabljeni čas za posamezno *Nalogo* iz *Seznama Nalog Iteracije*
K04	Učinkovitost dela	M05	WR (ang. Work Remaining) Preostali čas za posamezno *Nalogo* iz *Seznama Nalog Iteracije*
		M06	WS (ang. Work Spent) Porabljeni čas za posamezno *Nalogo* iz *Seznama Nalog Iteracije*
K05	Sprememba *Hitrosti*	M01	Skupno število predvidenih *Uporabniških zgodb* v *Iteraciji*
		M02	Skupno število potrjenaih *Uporabniških zgodb* v *Iteraciji*
K06	Napredek *Izdaje*	M03	Skupno število predvidenih *Uporabniških zgodb* v *Izdaji*
		M04	Skupno število potrjenaih *Uporabniških zgodb* v *Izdaji*
K07	Napredek *Iteracije*	M06	WS (ang. Work Spent) Porabljeni čas za posamezno *Nalogo* iz *Seznama Nalog Iteracije*
K08	Pogostost napak	M07	Velikost kode
		M08	Število napak, ugotovljenih na *Potrjevalnem sestanku Iteracije*
		M09	Število napak, poročanih v določenem obdobju po *Izdaji*
K09	Stroški ponovnega dela	M10	Porabljeni čas za popravke Nalog v *Seznamu Nalog Iteracije*
K10	Kvalitativna ocena zadovoljstva strank	M11	Rezultati ankete o zadovoljstvu strank na *Potrjevalnem sestanku Iteracije* (ocene so od 1 do 5, pri čemer je 1 najslabša ocena in 5 najboljša ocena)
K11	Obseg nadurnega dela	M06	Porabljeni čas za posamezno nalogo v *Seznamu Nalog Iteracije*
K12	Obseg sočasnega dela na več projektih	-	-
K13	Kvalitativna ocena delovnih pogojev	M12	Rezultati ankete o delovnih pogojih na *Retrospektivnem sestanku Iteracije* (ocene so od 1 do 5, pri čemer je 1 najslabša ocena in 5 najboljša ocena)
K14	BRE (točke zgodb)	M13	Dejanska zahtevnost *Uporabniške zgodbe*
		M14	Ocenjena zahtevnost *Uporabniške zgodbe*
K15	Povprečno število nerešenih *Ovir*	M15	Število nerešenih *Ovir* ob zaključku *Iteracije*
K16	Povprečen čas za reševanje *Ovir*	M16	Porabljeni čas za reševanje *Ovir*

Vrednosti osnovnih metrik pridobimo na rednih *Scrum* sestankih, kot je predstavljeno v Tabeli 30, razen metrike M09: Število napak, poročanih v določenem obdobju po *Izdaji* (za posamezno *Uporabniško zgodbo*), ki jo pridobimo po *Izdaji*.

Tabela 30: Zbiranje osnovnih metrik

Oznaka	Osnovna metrika	Sestanek
M01	Skupno število predvidenih *Uporabniških zgodb* v *Iteraciji*	*Načrtovalni sestanek Iteracije*
M02	Skupno število potrjenaih *Uporabniških zgodb* v *Iteraciji*	*Potrjevalni sestanek Iteracije*
M03	Skupno število predvidenih *Uporabniških zgodb* v *Izdaji*	*Načrtovalni sestanek Iteracije*
M04	Skupno število potrjenaih *Uporabniških zgodb* v *Izdaji*	*Potrjevalni sestanek Iteracije*
M05	WR (ang. Work Remaining) Preostali čas za posamezno *Nalogo* iz *Seznama Nalog Iteracije*	*Dnevni Scrum sestanek*
M06	WS (ang. Work Spent) Porabljeni čas za posamezno *Nalogo* iz *Seznama Nalog Iteracije*	*Dnevni Scrum sestanek*
M07	Velikost kode	*Retrospektivni sestanek Iteracije*
M08	Število napak, ugotovljenih na *Potrjevalnem sestanku Iteracije*	*Potrjevalni sestanek Iteracije*
M09	Število napak, poročanih v določenem obdobju po *Izdaji*	*Po izdaji*
M10	Porabljeni čas za popravke Nalog v *Seznamu Nalog Iteracije*	*Retrospektivni sestanek Iteracije*
M11	Rezultati ankete o zadovoljstvu strank na *Potrjevalnem sestanku Iteracije* (ocene so od 1 do 5, pri čemer je 1 najslabša ocena in 5 najboljša ocena)	*Potrjevalni sestanek Iteracije*
M12	Rezultati ankete o delovnih pogojih na *Retrospektivnem sestanku Iteracije* (ocene so od 1 do 5, pri čemer je 1 najslabša ocena in 5 najboljša ocena)	*Retrospektivni sestanek Iteracije*
M13	Dejanska zahtevnost *Uporabniške zgodbe*	*Retrospektivni sestanek Iteracije*
M14	Ocenjena zahtevnost *Uporabniške zgodbe*	*Načrtovalni sestanek Iteracije*
M15	Število nerešenih *Ovir* ob zaključku *Iteracije*	*Retrospektivni sestanek Iteracije*
M16	Porabljeni čas za reševanje *Ovir*	*Retrospektivni sestanek Iteracije*

Poleg navedenih metrik, bomo za izračun kazalnikov potrebovali parametre, navedene v Tabeli 31.

Tabela 31: Zbiranje ostalih podatkov

Parameter	Zbiranje podatkov	Oznaka
Število dni v *Iteraciji*	*Načrtovalni sestanek Iteracije*	SL (ang. Sprint Length)
Število članov *Razvojne skupine*	*Načrtovalni sestanek Iteracije*	NTM (ang. Number of Team Members)
Odstotek udeležbe članov *Razvojne skupine* pri projektu	*Načrtovalni sestanek Iteracije*	PTME (ang. Percentage of Team Member's Engagement in the Project)
Cena inženirske ure za posameznega člana *Razvojne skupine*	*Načrtovalni sestanek Iteracije*	CEH (ang. Cost of Engineering Hour)
Število pretečenih dni od začetka *Iteracije*	*Dnevni Scrum sestanek*	DE (ang. Days Elapsed)

Osnovne podatke, ki omogočajo izračun navedenih kazalnikov (z izjemo števila napak po izdaji) je možno pridobiti na sestankih, ki jih predpisuje metodologija *Scrum*, kar je skladno z načeli agilnosti.

Na *Dnevnem Scrum sestanku* se zbirajo podatki, ki jih opredeljuje izvorna različica metodologije *Scrum*: preostali čas, ki je potreben za potrjenaje posamezne *Naloge*, podatki o spremembi statusa *Naloge* ter podatki o *Ovirah*. Poleg navedenih je za potrebe modela potrebno na *Dnevnem Scrum sestanku* dodatno zbirati samo eno metriko: porabljeni čas na dan *d* za posamezno *Nalogo* v *Seznamu Nalog Iteracije*. Člani *Razvojne skupine* poleg ocene preostalega časa na dan *d* za posamezno *Nalogo* v *Seznamu Nalog Iteracije* poročajo tudi

porabljeni čas od prejšnjega *Dnevnega Scrum sestanku* oziroma število ur vloženega dela po posameznih *Nalogah* med dvema *Dnevnima Scrum sestankoma.*

Obseg administrativnega dela za zbiranje metrik pri modelu *AGIT* je enak obsegu administrativnega dela pri uporabi metodologije *Scrum*, povečan za zbiranje podatkov o porabljenem času. Menimo, da je načelo agilnosti ohranjeno in da navedena metrika ne predstavlja pomembno dodatno obremenitev razvijalcev.

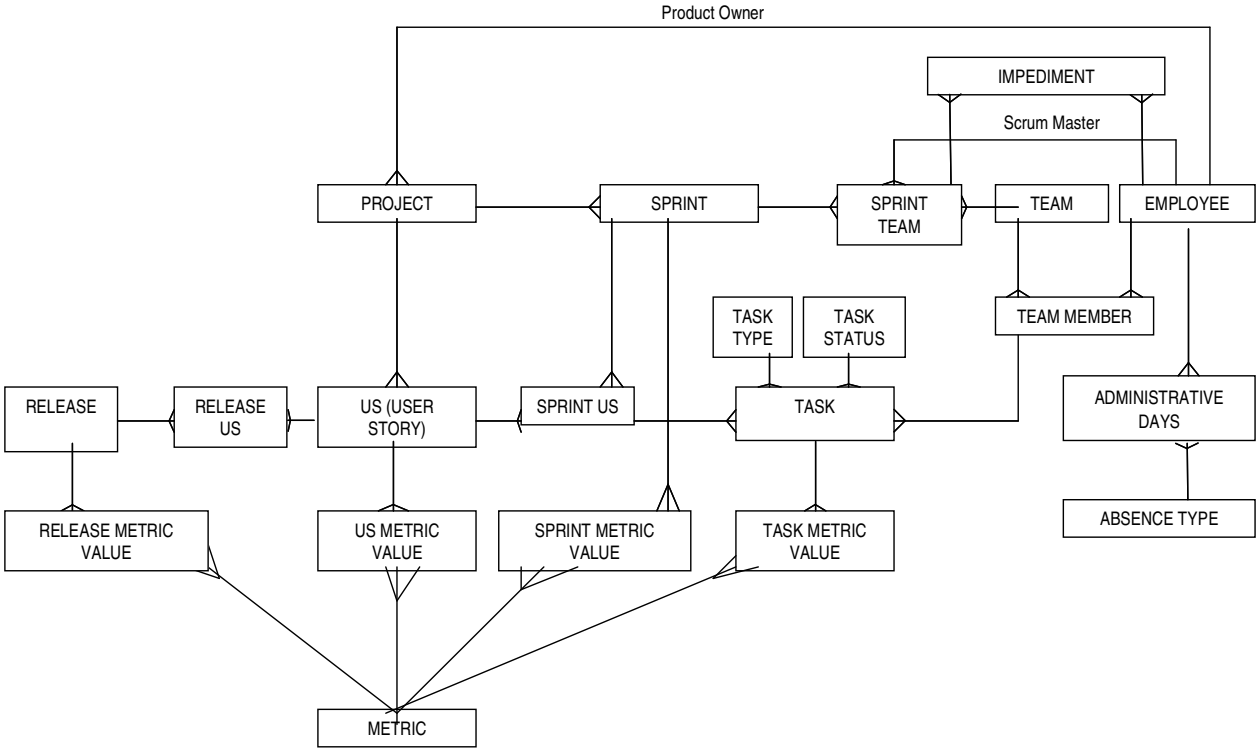
4.4.3 Podatkovni model

Podatkovni model repozitorija s podatki o razvojnem procesu, ki izpolnjuje zahteve modela *AGIT*, je bil prvič predstavljen leta 2007 [Mah07b]. S pomočjo shranjenih podatkov lahko analiziramo značilnosti razvojnega procesa.

V nadaljevanju bo predstavljena nadgradnja tega podatkovnega modela, v kateri bodo upoštevana spoznanja in izkušnje, pridobljene po letu 2007. Nazivi v podatkovnem modelu so v angleščini, zaradi lažje primerjave z rešitvami na tem področju.

Podatkovni model je predstavljen na Sliki 20 [Mah07b]. Model je generičen in omogoča vnos poljubnega števila kazalnikov, kar omogoča fleksibilnost glede dodajanja in umikanja posameznih kazalnikov iz nabora opazovanih kazalnikov. Predpogoj za uporabo modela je, da so rezultati vseh meritev istega tipa (npr. numerični tip). Imena in opisi kazalnikov so shranjeni v entiteti »Measure«, vrednosti kazalnikov pa so shranjene v tabelah »Measurement result« na ravni izdaje/Iteracije/Uporabniške zgodbe/naloge. Za vsako vrednost kazalnika se shranjuje podatek o datumu in času vnosa, kar omogoča, da lahko kadarkoli izračunamo kazalnike za izbrani datum in čas ter izdelamo zgodovinski pregled kazalnikov za poljubni časovni interval.

Slika 20: Podatkovni model



Osnovni model smo nadgradili s ciljnimi in mejnimi vrednostmi kazalnikov ter statusom *Nalog*.

V nadaljevanju so podane entitete za projekt (Tabela 32), *Izdajo* (Tabela 33) in meritve (Tabela 34).

Tabela 32: Entitete za projekt

Entiteta	Atributi
Project	(Project ID#, Project Description)
Employee	(Employee ID#, Employee Description)
Sprint Team	(Sprint ID#, Team ID#)
Team	(Team ID#, Team Description)
Team Member	(Team ID#, Employee ID#, Percentage of Engagement in the Project)
Administrative Days	(Employee ID#, Date#, Hours Not Worked, Absence Type ID#)
Absence Type	(Absence Type ID#, Absence Type Description)

Tabela 33: Entitete za Izdajo

Entiteta	Atributi
Release	(Release ID#, Release Description)
Release US	(Release US ID#, US ID#)
Sprint	(Sprint ID#, Sprint Description, Sprint Begin Date, Sprint End Date, Sprint Length, Sprint Estimated Date, Team ID#, Project ID#)
Sprint US	(Sprint ID#, US ID#, Sprint US Priority, Sprint US Status, Task ID#)
US	(US ID#, US Description, US Priority, US Category, US Status, Project ID#, Release ID#, Sprint ID#)
Task	(Task ID#, Task Description, Task Cost of Engineering Hour, Task Date, Task Active, Task Type ID#, Task Status ID#, US ID#, Sprint ID#, Team ID#, Employee ID#)
Task Status	(Task Status ID#, Task Status Description)
Task Type	(Task Type ID#, Task Type Description)
Impediment	(Impediment ID#, Impediment Description, Impediment Occurrence Date, Impediment Resolution Date, Sprint ID#, Team ID#, Employee ID#)

Za pravilen vnos so pomembni šifranti statusov *Uporabniških zgodb*, *Nalog Iteracije* in *Ovir.* Možne vrednosti statusov so npr. nekativen, v teku, potrjena, reaktiviran.

Tabela 34: Entitete za meritve

Entiteta	Atributi
Measure	(Measure ID#, Measure Name, Measure Description, Target value, Minimum value, Maximum value)
Release Measurement Result	(Release ID#, Measure ID#, Date#, Hour#, Measurement Result)
Sprint Measurement Result	(Sprint ID#, Measure ID#, Date#, Hour#, Measurement Result)
US Measurement Result	(US ID#, Measure ID#, Date#, Hour#, Measurement Result)
Task Measurement Result	(Task ID#, Measure ID#, Date#, Hour#, Measurement Result)

Meritve shranjujemo na ravni *Izdaje, Iteracije, Uporabniške zgodbe* in *Naloge Iteracije*.

Predstavljeni podatkovni model omogoča zbiranje in prikazovanje podatkov.

4.4.4 Prikazovanje podatkov

Doseganje ciljnih vrednosti je najlažje spremljati z uporabo grafov oziroma nadzornih plošč. Izvorna različica metodologije *Scrum* določa spremljanje *Padajočega diagrama napredka Izdaje* ter plošče s statusi *Uporabniških zgodb*.

Kazalniki modela *AGIT* so namenjeni predvsem *Skrbniku Metodologije Scrum,* ki komunicira z vsemi ostalimi deležniki in pri poročanju statusa projekta lahko uporabi kazalnike, ki so primerni za posameznega deležnika. Poleg poročanja je namen spremljanja kazalnikov tudi ocena doseganja roka in pravočasno ukrepanje. Pri tem je pomembna izvedba »What if« analize oziroma različnih scenarijev, ki omogočajo *Skrbniku Metodologije Scrum,* da lažje sprejme odločitev glede ukrepanja. S simulacijo različnih ukrepov (npr. povečanje hitrosti tima, zmanjšanje števila *Uporabniških zgodb,* sprememba roka) lahko hitro ugotovi vpliv posamezne spremembe in ga tudi na transparenten in enostaven način predstavi zainteresiranim deležnikom.

Skrbnik Metodologije Scrum lahko za prikazovanje podatkov uporablja Excel preglednico in grafe, bolj Uporabniško prijazna pa je uporaba nadzornih plošč, na katerih je prikazanih več grafov. Tako lahko istočasno spremlja vpliv spremembe na več ravneh: raven *Naloge, Uporabniške zgodbe, Iteracije in Izdaje.*

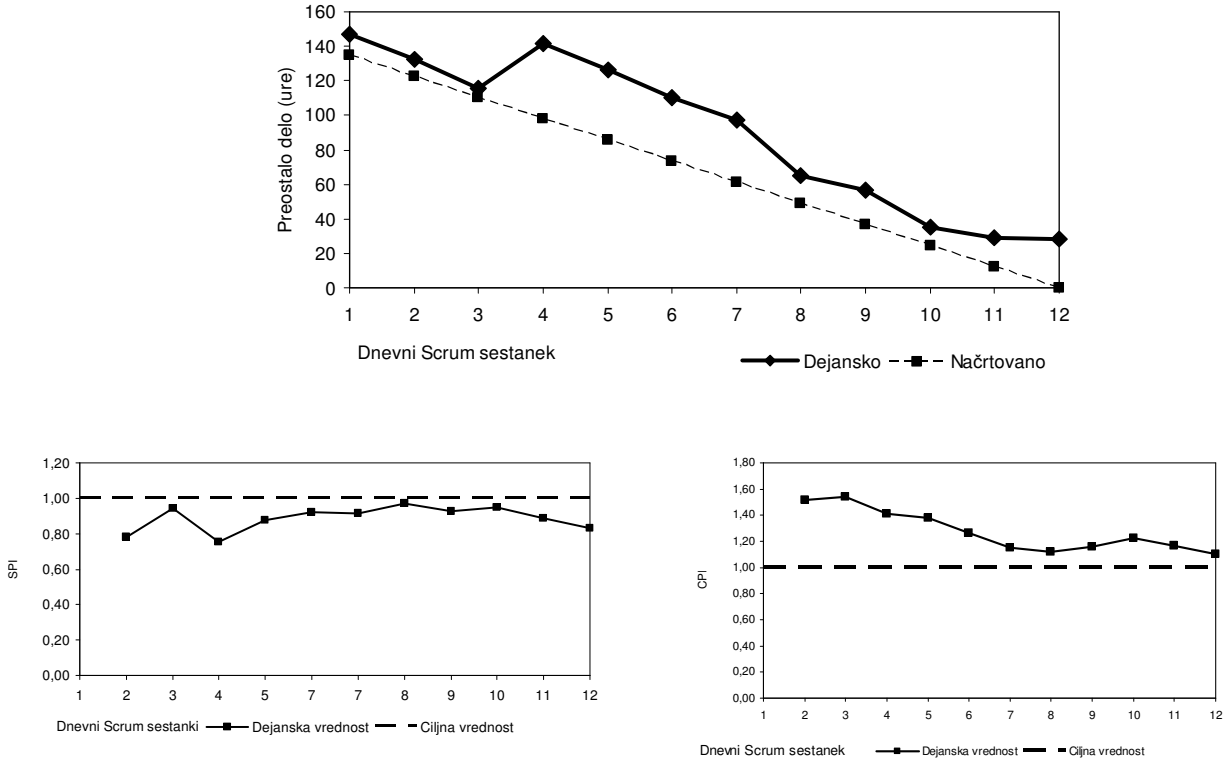
Kazalnike modela *AGIT* lahko prikažemo z uporabo naslednjih diagramov:

- Diagram hitrosti, ki prikazuje število točk *Uporabniških zgodb* iz *Seznama Uporabniških zgodb*, ki so bile potrjene v posamezni *Iteraciji;*
- *Padajoči diagram spremljanja napredka Izdaje*, ki prikazuje preostali obseg dela na začetku vsake *Iteracije*, izražen v seštevku točk nepotrjenih *Uporabniških zgodb* v *Seznamu Uporabniških zgodb;*
- *Padajoči diagram spremljanja napredka Iteracije,* ki prikazuje preostali obseg dela na začetku vsakega *Dnevnega Scrum sestanka*, izražen v seštevku ur potrebnih za dokončanje *Nalog* v *Seznamu Nalog Iteracije.*

Zgoraj navedene diagrame bomo uporabili v poglavju o študiji primera.

Zgled nadzorne plošče smo podali na sliki 21.

Slika 21. Zgled AGIT nadzorne plošče za prvo Iteracijo



5. ŠTUDIJ PRIMERA UPORABE MODELA *AGIT*

5.1 Uvod

5.1.1 Predstavitev študije primera

Študija primera je bila del predhodnih raziskav Laboratorija za tehnologijo programske opreme, kar je opisano v uvodnem delu četrtega poglavja. Za razliko od navedenih prispevkov, v katerih so prikazani rezultati izračuna izbranih kazalnikov za izbrane *Iteracije*, bodo v nadaljevanju podani rezultati kazalnikov, ki so pomembni za projektno vodenje, za vse *Iteracije* ter analiza teh rezultatov.

Leta 2011 je bil model *AGIT* preizkušen tudi v gospodarskem okolju. V največji slovenski časopisni hiši so uporabili metodologija *Scrum* pri projektu prenove spletne izdaje dnevnika z najvišjo naklado v Sloveniji. Vodja projektnega tima je bil eden izmed študentov iz generacije 2008/2009, pri kateri je bilo na vajah prvič uporabljeno metodologija *Scrum* ter model *AGIT*. Opis uporabe metodologije *Scrum* in modela *AGIT* je podan v [Ure12]. Dogovorjeno je bilo, da avtor modela *AGIT* pomaga pri pripravi na uvedbo metodologije *Scrum* ter sodeluje kot opazovalec pri prvih sedmih *Iteracijah*. Rezultati so opisani v [Mah12d] in [Mah12c]. Podatki, pridobljeni tekom projekta, so bili uporabljeni pri študiji primera, opisani v nadaljevanju.

5.1.2 Predstavitev podjetja

Obravnavana časopisna hiša je ena vodilnih in vplivnejših na slovenskem medijskem trgu. Njeno temeljno poslanstvo je javnosti posredovati čim širši spekter medijskih vsebin. Kot verodostojen in relevanten vir informacij želi na čim več področjih postati vodilni mnenjski voditelj, ki mu bodo ljudje zaupali, ter s pomočjo pridobljenih informacij sprejemali vse svoje odločitve, kar jim bo omogočalo lažje in srečnejše sobivanje.

Vizija obravnavane časopisne hiše je postati vodilna medijska hiša, ki bo s ponudbo vsebin ter proaktivnim odzivanjem na tehnološke, tržne in družbene spremembe z vsemi svojimi deležniki gradila dolgoročne odnose in zadovoljevala vse njihove potrebe. Med deležnike oz. ključno ciljno javnost uvrščajo zaposlene, bralce, naročnike, oglaševalce, prodajno osebje, distributerje, investitorje, medijske zakupnike in medije.

Časopisna hiša ponudbo produktov in storitev nenehno prilagaja, dopolnjuje in nadgrajuje, s čimer se želi čim hitreje in aktivneje odzivati na spreminjajoče se tržne zahteve ter izpolniti potrebe, želje in pričakovanja svojih bralcev, oglaševalcev in poslovnih partnerjev. Družba ponuja široko ponudbo vsebin s katero dosega najrazličnejše segmente bralcev na tiskanih, spletnih, mobilnih in tabličnih platformah. Raznolikost platform omogoča, da so vsebine uporabnikom na voljo 24 ur na dan, 7 dni v tednu, vsak dan v letu, pri čemer se bralci lahko sami odločajo kdaj in kje si želijo interaktivnosti s posameznimi mediji in edicijami.

Leta 2010 je časopisna hiša postala članica kluba najboljših časopisov INCQC (ang. International Newspaper Color Quality Club). Njen dnevnik je po zaslugi oddelka za razvoj digitalnih platform postal prvi slovenski časopis na priljubljenem tabličnem računalniku iPad.

Podjetje se je istega leta odločilo za prenovo spletnih vsebin in izvedlo nekaj manjših projektov prenove prilog k rednim izdajam. Pri tovrstnih projektih je bil razvojni tim sestavljen iz zaposlenih v oddelku za razvoj spletnih vsebin in zunanjih razvijalcev.

Leta 2011 se je podjetje odločilo za prenovo spletne izdaje dnevnika, katerega tiskana izdaja je med najbolj branimi v državi in je bil leta 2010 izbran kot najbolj priljubljena znamka med dnevnimi časopisi (Trusted Brand). V ta namen je bil odobren projekt agilnega razvoja časopisnega portala z uporabo metodologije *Scrum*, ki je opisan v nadaljevanju.

5.1.3 Predstavitev projekta

Časopisna hiša je s projektom »Agilni razvoj časopisnega portala z uporabo metodologije *Scrum*« (v nadaljevanju *Scrum* projekt) želela zagotoviti konkurenčnost portala in optimizirati delovni proces.

Namenski cilji *Scrum* projekta so bili:
- povečati število uporabnikov in njihovo angažiranost na portalu;
- urednikom olajšati delo s portalom in zagotoviti stabilne temelje za nadaljnji razvoj portala;
- zagotoviti enotno tehnološko platformo spletnih mest ter metodologijo razvoja, ki bo omogočala standardizacijo procesov razvoja in vzdrževanja.

Objektni cilji *Scrum* projekta so bili:
- oblikovno in vsebinsko prenoviti časopisni portal;
- preseliti portal na odprtokodno platformo Drupal in
- pilotno uvesti metodologija *Scrum* v delovni proces oddelka spletnega razvoja.

Obstoječi portal je bil postavljen na interno razviti platformi. V okviru *Scrum* projekta je bil izveden prehod na odprtokodni sistem za upravljanje vsebin in ogrodje za upravljanje vsebin Drupal. Skupnost Drupal.org šteje več kot 750.000 članov, med katerimi so nekateri tudi uporabniki metodologije *Scrum.* Med referenčne uporabnike Drupala sodi tudi spletna izdaja revije The Economist.com, ki je bila razvita z uporabo metodologije *Scrum*.

Metodologija *Scrum* je bila izbrana zaradi spremenljivih zahtev, narave projekta in kratkih časovnih rokov. Za podporo administrativnemu delu je bilo uporabljeno orodje za vodenje agilnih projektov »Agilo for *Scrum*«.

Projektna skupina je vključevala zaposlene v podjetju, zunanje izvajalce in vodjo Laboratorija za tehnologijo programske opreme, ki je imel vlogo opazovalca. Število članov *Razvojne skupine* je nihalo med šest in osem. Zunanji izvajalci so bili dislocirani, zato so *Dnevne Scrum sestanke* izvajali z uporabo spletnih tehnologij.

Scrum projekt je potekal od maja do novembra 2011, skupaj sedem mesecev. Pred pričetkom projekta so bile izvedene pripravljalne aktivnosti.

5.1.4 Predstavitev podmnožice metrik modela *AGIT*

Analiza podatkov je ključnega pomena pri zagotavljanju učinkovitosti, ker omogoča ugotavljanje odstopanj, vzrokov za odstopanja ter odpravljanje le teh. Pri analizi podatkov ima najbolj pomembno vlogo *Skrbnik Metodologije Scrum,* katerega cilj je zagotoviti nemoten potek projekta z učinkovitim odpravljanjem *Ovir,* pri čemer lahko *Ovire* razumemo v širšem pomenu in sicer kot odstopanja od zastavljenih ciljev.

V Tabeli 35 je navedena tudi prioriteta metrike iz vidika projektnega vodenja, ki je lahko visoka, srednja ali nizka. Visoka prioriteta pomeni, da je metrika zelo pomembna za zagotavljanje učinkovitosti *Scrum* projekta oziroma da je obvezna zaradi zakonskih ali drugih zahtev. Srednja prioriteta pomeni, da je metrika pomembna za zagotavljanje učinkovitosti *Scrum* projekta, vendar ni nujna. Nizka prioriteta pomeni, da je metrika manj pomembna za zagotavljanje učinkovitosti *Scrum* projekta.

Tabela 35: Prioritete metrik iz vidika projektnega vodenja

ZŠ	Kazalnik	Prioriteta
K01	Doseganje obsega	Srednja
K02	Časovni indeks	Visoka
K03	Stroškovni indeks	Visoka
K04	Učinkovitost dela	Srednja
K05	Sprememba *Hitrosti*	Visoka
K06	Napredek *Izdaje*	Visoka
K07	Napredek *Iteracije*	Visoka
K08	Pogostost napak	Srednja
K09	Stroški ponovnega dela	Srednja
K10	Kvalitativna ocena zadovoljstva strank (1-5)	Srednja
K11	Obseg nadurnega dela	Srednja
K12	Obseg sočasnega dela na več projektih	Srednja
K13	Kvalitativna ocena delovnih pogojev (1-5)	Srednja
K14	BRE (točke zgodb)	Visoka
K15	Povprečno število nerešenih ovir	Srednja
K16	Povprečen čas za reševanje ovir	Srednja

V Tabeli 36 so prikazane izpeljane metrike z visoko prioriteto z vidika vodenja projekta. Primarni cilj vodenja projekta je zagotavljanje uspešnosti in učinkovitosti projekta, zato je pomembno, da je dnevno seznanjen s ključnimi kazalniki.

Tabela 36: Podmnožica metrik modela AGIT

ZŠ	Kazalnik	Raven
K02	Časovni indeks	*Iteracija*
K03	Stroškovni indeks	*Iteracija*
K05	Sprememba *Hitrosti*	*Izdaja*
K06	Napredek *Izdaje*	*Izdaja*
K07	Napredek *Iteracije*	*Iteracija*
K14	BRE (točke zgodb)	*Iteracija*

Predpostavka modela je, da *Skrbnik Metodologije Scrum* z dnevno analizo zgoraj navedenih kazalnikov lahko pravočasno ugotovi odstopanja od zastavljenih ciljev in sproži ustrezne ukrepe.

5.2 Opis primera

5.2.1 Časovni vidik

Dolžina *Iteracije*

Izbrana dolžina *Iteracije* je bila tri koledarske tedne oziroma 21 koledarskih dni, kar je pri večini *Iteracij* pomenilo 14 delovnih dni.

Iteracija se je pričela v četrtek prvega tedna z *Načrtovalnim sestankom Iteracije*, končala pa tretji teden v torek, s *Potrjevalnim sestankom Iteracije* in *Retrospektivnim sestankom Iteracije*. Od skupno 14 dni je bilo 12 dni namenjeno razvoju programske opreme, v teh dneh so *Skrbnik Metodologije Scrum* in *Razvojna skupina* imeli redne 15-minutne *Dnevne Scrum sestanke*, ki so bili izvedeni z uporabo audio spletne konference, zaradi dislociranosti članov *Razvojne skupine*.

Potek projekta je razviden iz Tabele 37.

Tabela 37: Potek projekta

Sprint	Začetek	Konec	Število delovnih dni	Opomba
Sprint 1	11.5.2011 (sreda)	31.5.2011 (torek)	15	
Sprint 2	2.6.2011 (četrtek)	21.6.2011 (torek)	14	
Sprint 3	23.6.2011 (četrtek)	12.7.2011 (torek)	14	
Sprint 4	14.7.2011 (četrtek)	2.8.2011 (torek)	14	
Sprint 5	4.8..2011 (četrtek)	30.8.2011 (torek)	17	Dopusti
Sprint 6	1.9.2011 (četrtek)	20.9.2011 (torek)	14	
Sprint 7	22.9.2011 (četrtek)	11.10.2011 (torek)	14	
Sprint 8	13.10.2011 (četrtek)	8.11.2011 (torek)	17	Prazniki
Sprint 9	9.11.2011 (sreda)	29.11.2011 (torek)	15	

Ocenjevanje zahtevnosti *Uporabniških zgodb*

Za ocenjevanje zahtevnosti *Uporabniških zgodb* je bila izbrana metoda *Načrtovalni poker*. Skladno s to metodo je ocena izražena s številom točk, pri čemer so dopustne vrednosti 0.5, 1, 2, 3, 5, 8, 13 in 20. Dogovorjeno je bilo, da ena točka predstavlja en človek-dan oziroma šest ur. Ocene zahtevnosti so podane v Tabeli 38.

Tabela 38: Ocenjevanje zahtevnosti Uporabniških zgodb

ZŠ	Zahtevnost *Uporabniške zgodbe* (točke)	Zahtevnost *Uporabniške zgodbe* (človek-dni)	Zahtevnost *Uporabniške zgodbe* (ure)
1	0.5 točk	0.5 človek-dni	3 ure
2	1 točka	1 človek-dan	6 ur
3	2 točki	2 človek-dneva	12 ur
4	3 točke	3 človek-dnevi	18 ur
5	4 točke	4 človek-dnevi	24 ure
6	8 točk	8 človek-dni	48 ur
7	13 točk	13 človek-dni	78 ur
8	20 točk	20 človek-dni	120 ur

Začetna hitrost *Razvojne skupine*
Pri načrtovanju *Iteracije* je pomemben podatek hitrost *Razvojne skupine*, ki se nanaša na število točk realiziranih v *Iteraciji.*
Razvojna skupina je prvič sodelovalo pri *Scrum* projektu, zato je bila začetna hitrost ocenjena kot produkt števila delovnih dni (14), števila razvijalcev (5) in uteži, ki je predstavljala izkušenost posameznega razvijalca, skupaj 32,5 točk. V naslednjih *Iteracijah* so bili uporabljeni podatki o dejansko doseženi hitrosti, pri čemer so bile upoštevane točke *Uporabniških zgodb,* ki so imele status »potrjena« (»sprejeta«) ali »potrjena s pridržkom« (»Sprejeta s pridržkom«).

5.2.2 Organizacijski vidik

***Scrum* vloge**
Uredništvo je za *Produktnega vodje* določilo pomočnika odgovorne urednice, ki je v preteklosti že sodeloval pri projektih razvoja večjih spletnih mest. Za vlogo *Skrbnika Metodologije Scrum* sta bila imenovana vodja razvoja in njegova pomočnica, zaradi velikega obsega njunih dodatnih zadolžitev (komunikacija in usklajevanje razvoja z ostalimi uredništvi v hiši na drugih projektih). Izkazalo se je, da bi bilo bolj primerno, če bi imela vlogo *Skrbnika Metodologije Scrum* samo ena oseba.
Razdelitev *Scrum* vlog je razvidna iz Tabele 39.

Tabela 39: Scrum vloge

Vloga	Nosilec	Opomba
Produktni vodja	Pomočnik odgovorne urednice	
Skrbnik Metodologije Scrum	Vodja razvoja in njegova pomočnica	Odstopanje od metodologije Scrum
Razvojna skupina	Notranji razvijalci Zunanji razvijalci	

Usposabljanje
V časopisni hiši je bilo metodologija *Scrum* uporabljeno prvič, zato je bilo za vse člane projektne skupine izvedeno izobraževanje, na katerem je bila predstavljeno metodologija *Scrum* in pričakovanja glede posameznih vlog in njihovih nalog.

Sodelovanje z *Produktnim vodjo*
Pri metodologiji *Scrum* je vloga *Produktnega vodje* ključna za doseganje ciljev projekta, zato je pomemben partnerski odnos med *Produktnim vodjo*, *Skrbnikom Metodologije Scrum* in *Razvojno skupino.*
Produktni vodja je bil prisoten na vseh *Načrtovalnih sestankih Iteracije* in *Potrjevalnih sestankih Iteracije.* Na začetku je *Produktni vodja* sodeloval v obsegu kot pri običajnih projektih. Postopoma se je obseg njegovega dela povečeval in postajal skladen z zahtevami metodologije *Scrum* glede vzdrževanja *Seznama Uporabniških zgodb* in izvajanja sprejemnih testov. Od kakovosti testiranja s strani *Produktnega vodje* je v veliki meri odvisna kakovost končnega izdelka, zato je pomembno, da *Produktni vodja* načrtuje dovolj časa za testiranje.

Sestava *Razvojne skupine*

Na začetku projekta so v *Razvojni skupini* bili trije razvijalci. Dva razvijalca sta bila dislocirana, tretji razvijalec je delal na sedežu podjetja. Dva razvijalca sta imela izkušnje z novo platformo, en razvijalec je bil začetnik.

V peti *Iteraciji* sta se *Razvojni skupini* pridružila dva razvijalca: en zunanji in en notranji. Novi Zunanji razvijalec je istočasno sodeloval tudi pri drugih projektih. Njegovi nalogi sta bili razrez oblikovnih predlog in izdelava teme portala, kar je zahtevalo specifična znanja, ki jih drugi člani *Razvojne skupine* niso imeli. Izkazalo se je, da se zunanji razvijalec ni mogel udeleževati *Dnevnih Scrum sestankov*, kar je negativno vplivalo na dinamiko tima. Novi notranji razvijalec je prav tako istočasno sodeloval tudi pri drugih projektih in ni imel izkušenj z novo platformo. Čas, ki je bil potreben za njegovo uvajanje, je negativno vplival na hitrost tima.

V šesti *Iteraciji* se je *Razvojni skupini* pridružil zunanji razvijalec, ki je sodeloval samo na tem projektu in je delal na dislocirani lokaciji. Imel je veliko izkušenj z novo platformo in kmalu je dosegel produktivnost ostalih članov *Razvojne skupine*.

V sedmi *Iteraciji* je bil notranji razvijalec, ki se je pridružil *Razvojni skupini* v peti *Iteraciji,* prerazporejen na drugi projekt.

Do zaključka izdaje je *Razvojna skupina* štelo pet članov in njegova sestava se ni več spreminjala. Sestava *Razvojne skupine* je razvidna iz Tabele 40.

Tabela 40: Sestava Razvojne skupine

Sprint	Začetek	Konec	Število razvijalcev	Opomba
Sprint 1	11.5.2011 (sreda)	31.5.2011 (torek)	3	
Sprint 2	2.6.2011 (četrtek)	21.6.2011 (torek)	3	
Sprint 3	23.6.2011 (četrtek)	12.7.2011 (torek)	3	
Sprint 4	14.7.2011 (četrtek)	2.8.2011 (torek)	3	
Sprint 5	4.8..2011 (četrtek)	30.8.2011 (torek)	5	2 nova razvijalca
Sprint 6	1.9.2011 (četrtek)	20.9.2011 (torek)	6	1 nov razvijalec
Sprint 7	22.9.2011 (četrtek)	11.10.2011 (torek)	5	1 nov razvijalec iz 5. Iteracije prerazporejen

5.2.3 Vidik kakovosti

Izbrani kazalniki učinkovitosti

Tekom projekta so bili za merjenje učinkovitosti uporabljeni naslednji kazalniki:

- ocene zahtevnosti *Uporabniških zgodb* (v točkah);
- ocene hitrosti *Razvojne skupine* (v točkah),
- obseg porabljenega dela na posameznih *Nalogah* (v urah) in
- ocene preostalega dela na posameznih *Nalogah* (v urah).

Po zaključku projekta so bili zbrani podatki uporabljeni za preizkus modela *AGIT*. Cilj preizkusa je bil ugotoviti ali bi lahko z uporabo dodatnih informacij, ki jih omogoča model *AGIT*, izboljšali učinkovitost projekta. Pri tem so bili uporabljeni naslednji kazalniki:

- Hitrost *Razvojne skupine* (v točkah);
- *Padajoči diagram spremljanja napredka Izdaje* (v točkah);
- Analiza točnosti ocen;
- *Padajoči diagram spremljanja napredka Iteracije* (v urah);
- Stroškovni indeks;
- Časovni indeks.

Kriterij za potrjevanje Uporabniške zgodbe

Potrjena *Uporabniška zgodba* mora biti neposredno uporabna v produkcijskem okolju. Pred začetkom projekta je potrebno opredeliti kriterij za potrjevanje *Uporabniške zgodbe* oziroma pogoje, ki jih mora *Uporabniška zgodba* izpolnjevati, da ji *Produktni vodja* lahko dodeli enega izmed naslednjih statusov: »potrjena« (ang. »done«), »potrjena s pridržkom« ali »zavrnjena«.

Status »potrjena« pomeni, da je delo na *Uporabniški zgodbi* zaključeno. Status »potrjena s pridržkom« pomeni, da so potrebni manjši popravki. Status »zavrnjena« pomeni, da bo *Uporabniška zgodba* izločena ali prenesena v naslednjo *Iteracijo*. Kriteriji za zaključek *Uporabniške zgodbe* so razvidni iz Tabele 41.

Tabela 41: Kriteriji za zaključek Uporabniške zgodbe

ZŠ	Pogoj	Nosilec	Potrjevalec
1	Uspešno opravljeni sprejemni testi	*Produktni vodja*	
2	Koda je napisana v skladu s standardi kodiranja	*Član Razvojne skupine*	*Skrbnik Metodologije Scrum*
3	Koda je ustrezno komentirana	*Član Razvojne skupine*	*Skrbnik Metodologije Scrum*
4	Obstaja dokumentacija o delovanju in namestitvi	*Član Razvojne skupine*	*Skrbnik Metodologije Scrum*
5	Rešitev je pregledana s strani drugega razvijalca	*Član Razvojne skupine*	*Skrbnik Metodologije Scrum*
6	Rešitev je pregledana s strani *Skrbnika Metodologije Scrum*	*Skrbnik Metodologije Scrum*	
7	Rešitev je pregledana in dokončno potrjena s strani *Produktnega vodje*	*Produktni vodja*	

Orodje za vodenje agilnih projektov

Ključni dejavnik pri izbiri orodja za podporo administrativnemu delu projekta je bila dislociranost zunanjih razvijalcev, zaradi katere klasična tabla, na katero razvijalci lepijo lističe z zgodbami in na kateri je narisan burndown diagram, ni prišla v poštev. Izbrano je bilo orodje za vodenje agilnih projektov, ki temelji na spletnem vmesniku »Agilo for *Scrum*«. Z orodjem so razvijalci lahko upravljali *Seznam Uporabniških zgodb* in *Seznam Nalog Iteracije* ter vnašali preostali in porabljeni čas za posamezno *Nalogo*. Funkcije orodja za projektno vodenje so razvidne iz Tabele 42.

Tabela 42: Funkcije orodja za projektno vodenje

ZŠ	Funkcija	Vnašalec	Uporabnik
1	Upravljanje *Seznama Uporabniških zgodb*	*Produktni vodja*	Vsi
2	Vnos preostalega časa za posamezno *Nalogo*	Razvijalec	Vsi
3	Vnos porabljenega časa za posamezno *Nalogo*	Razvijalec	Vsi

Zbiranje podatkov

Tekom projekta je *Razvojna skupina* zbiral podatke, ki so osnova za izračun kazalnikov učinkovitosti. Na dnevnih *Scrum* sestankih so bili z uporabo orodja Agilo evidentirani podatki

o preostalem in porabljenem času za vsako nalogo ter podatki o statusu nalog ter Uporabniških zgodb.
Pri evidentiranju podatkov je bilo potrebno upoštevati omejitve orodja glede vnosa ocen. V osnovnem pregledu so bili vidni podatki o zadnjem vnosu, preostali vnosi pa so bili zabeleženi v dnevniku vnosov. Tovrstno evidentiranje je bilo primerno za spremljanje učinkovitosti tekočega dne, ni pa omogočalo analize za obdobje od začetka Sprinta. Za potrebe izračuna kazalnikov učinkovitosti na ravni dnevnega *Scrum* sestanka je bilo potrebno iz dnevnika vnosov rekonstruirati podatke o preostalem in porabljenem času za posamezni dnevni *Scrum* sestanek. Pri tem je bilo uporabljeno orodje Excel.
Na dnevnih *Scrum* sestankih so se vnašale kumulativne vrednosti porabljenega časa. Za potrebe izračuna kazalnikov učinkovitosti je bilo potrebno izračunati razliko med kumulativnimi vrednostmi. Znesek porabljenega časa za posamezni dnevni *Scrum* sestanek je bila potem vnešen v Excel preglednico.
Ob rekonstrukciji podatkov je bilo potrebno upoštevati tudi čas vnosa. Nekateri podatki so bili vneseni pred dnevnim *Scrum* sestankom, na samem sestanku pa so bili lahko tudi večkrat spremenjeni. Zato je bilo pri rekonstrukciji upoštevano pravilo, da se podatki, ki so vneseni po 17. uri delovnega dne evidentirajo za prvi naslednji delovni dan. V primeru večkratnih vnosov istega dne, ki so si sledili v razmiku manjšem od ene ure, je bila upoštevana le zadnja vrednost vnosa.
Nekatere naloge so bile dodane v Product Backlog med Sprintom, npr. ob razdelitvi večjih nalog na manjše. V takih primerih je bila ocena preostalega časa enaka nič na vseh dnevnih *Scrum* sestankih do tistega dnevnega *Scrum* sestanka, na katerem je bila naloga dodana.
Poleg navedenih aktivnosti je bilo potrebno nekatere podatke dodatno prilagoditi, pri čemer je bilo možnih več rešitev.

Prikazovanje podatkov
Tekom projekta je *Razvojna skupina* lahko z uporabo orodja Agilo vzdrževal Sprint backlog ter v vsakem trenutku izrisal Burndown diagram.
Pri preizkusu modela *AGIT* je za izračun in prikaz kazalnikov merjenja učinkovitosti bila uporabljena Excel preglednica.

5.2.4 Potek *Iteracije*

Načrtovalni sestanek Iteracije
V četrtek prvega tedna *Iteracije* se je projektna skupina zbrala na isti fizični lokaciji in izvedla *Načrtovalni sestanek Iteracije.*
Sestanek je potekal v dveh delih:

- V prvem delu sestanka je *Produktni vodja* predstavil *Uporabniške zgodbe* z najvišjo prioriteto in bil na razpolago *Razvojni skupini* za dodatne informacije. Potem je *Razvojna skupina* ocenilo zahtevnost izbranih *Uporabniških zgodb. Produktni vodja* je nato izbral ustrezno število *Uporabniških zgodb*, pri čemer je število točk moralo biti manjše ali enako kapaciteti *Razvojne skupine*, ki je enaka produktu *Hitrosti* in števila delovnih dni.
- V drugem delu sestanka je *Razvojna skupina* razdelilo *Uporabniške zgodbe* na *Naloge* in ocenilo število ur, potrebnih za realizacijo *Naloge* ter določilo nosilca *Naloge*.

Končni rezultat *Načrtovalnega sestanka Iteracije* je *Seznam Nalog Iteracije,* ki vsebuje *Uporabniške zgodbe* in njim podrejene *Naloge*, predvidene za realizacijo v tekoči *Iteraciji.*

Dnevni Scrum sestanek

Ob 9. uri v petek prvega tedna so *Skrbnik Metodologije Scrum* in *Razvojna skupina* izvedli prvega od 12 *Dnevnih Scrum sestankov,* ki so trajali 15 minut in so potekali na način spletne audio konference.

Vsak razvijalec je povedal, kaj je naredil prejšnji dan, kaj bo naredil isti dan in ali potrebuje pomoč *Skrbnika Metodologije Scrum* pri reševanju *Ovir*. Največkrat je pomoč bila potrebna zaradi vsebinskih vprašanj, v takšnih primerih se je *Skrbnik Metodologije Scrum* dogovoril z *Produktnim vodjo* za poznejši sestanek. Vsak razvijalec je za svoje *Naloge* s pomočjo orodja Agilo zabeležil, koliko ur dela je opravil in ocenil število ur preostalega dela do dokončne realizacije *Naloge*.

V primerih, ko je razvijalec zaključil *Uporabniško zgodbo*, je *Skrbnik Metodologije Scrum* z uporabo kriterija za zaključevanje Uporabniških zgodb sprejel odločitev o izvedbi sprejemnega testa s strani *Produktnega vodje*, ki je sprejel končno odločitev o potrditvi *Uporabniške zgodbe.*

Potrjevalni sestanek Iteracije

V torek tretjega tedna *Iteracije* se je projektna skupina zbrala na isti lokaciji in izvedla *Potrjevalni sestanek Iteracije*, na katerem je *Produktni vodja* določil status predstavljenih Uporabniških zgodb: »potrjena«, »potrjena s pridržkom« ali »zavrnjena«.

Retrospektivni sestanek Iteracije

Po *Potrjevalnem sestanku Iteracije* so *Skrbnik Metodologije Scrum* in *Razvojna skupina* isti dan izvedli tudi *Retrospektivni sestanek Iteracije.*

Na *Retrospektivnem sestanku Iteracije* je vsak sodelujoči povedal svoje mnenje o tem, kaj je v pretekli *Iteraciji* bilo dobro narejeno, kaj bi lahko bilo narejeno boljše ter nove prakse, ki bi jih lahko preizkusili.

Rezultat *Retrospektivnega sestanka Iteracije* je bil skupni seznam predlogov za izboljšave, iz katerega so izbrali predloge, ki bodo realizirani v naslednji *Iteraciji*.

Potek *Iteracije* je razviden iz Tabele 43.

Tabela 43: Potek Iteracije

Dogodek	Dan	Trajan je	Kraj	Rezultat
Načrtovalni sestanek Iteracije	Četrtek prvega tedna	6 ur	Ista fizična lokacija	*Seznam Nalog Iteracije*
Dnevni Scrum sestanek	Vsak dan ob 9:00h	15 minut	Spletna audio konferenca	Statusi *Nalog*, ocene preostalega časa in podatki o porabljenem času
Potrjevalni sestanek Iteracije	Torek tretjega tedna	6 ur	Ista fizična lokacija	Potrjene *Uporabniške zgodbe*
Retrospektivni sestanek Iteracije	Torek tretjega tedna	6 ur	Ista fizična lokacija	Seznam predlogov za izboljšave
Vzdrževalne aktivnosti	Sreda tretjega tedna	6 ur	Različne fizične lokacije	Tehnična dokumentacija, tehnične priprave

5.2.6 Rezultati projekta

Po zaključku projekta je časopisni portal povečal svoj doseg. Rezultati merjenja obiskanosti slovenskih spletnih strani Moss, da je pred prenovo (oktober 2011) spletno stran mesečno obiskalo 200.000 bralcev, po prenovi (december 2011) pa skoraj 240.000. Zadnji podatki o številu obiskov (oktober 2012) uvrščajo časopisni portal na 16. mesto med medijskimi portali v Sloveniji. Skladno s strategijo podjetja časopisni portal nenehno posodabljajo, vsebinsko nagrajujejo in dodajajo nove, uporabnikom prijazne možnosti.

Izkušnje pri izvedbi projekta so opisane v [Ure12]. Član *Razvojne skupine* in avtor diplomskega dela se je v šolskem letu 2008/2009 pri praktičnem delu poučevanja predmeta Tehnologija programske opreme na Fakulteti za računalništvo in informatiko Univerze v Ljubljani udeležil projekta z uporabo metodologije *Scrum.* Vodja Laboratorija za tehnologijo programske opreme in mentor diplomskega dela je sodeloval pri projektu kot opazovalec pri prvih sedmih *Iteracijah*.

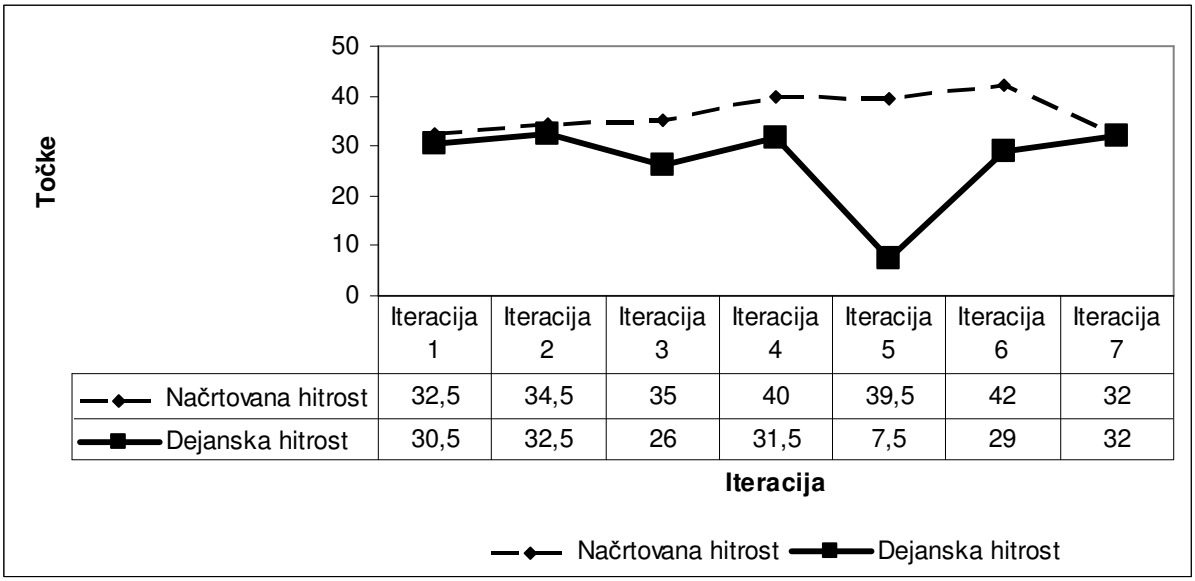
5.3 Spremljanje učinkovitosti *Izdaje*

5.3.1 Spremljanje *Hitrosti*

Razvojna skupina je spremljalo načrtovano in dejansko *Hitrost.* Ocene *Hitrosti* za posamezno *Iteracijo* so bile narejene na *Načrtovalnem sestanku Iteracije*. Na koncu *Iteracije* je bila izračunana dejanska *Hitrost* kot seštevek točk *Uporabniških zgodb,* ki jih je potrdil *Produktni vodja.*

Na Sliki 22 so podane spremembe *Hitrosti.*

Slika 22: Spremembe Hitrosti



	Iteracija 1	Iteracija 2	Iteracija 3	Iteracija 4	Iteracija 5	Iteracija 6	Iteracija 7
Načrtovana hitrost	32,5	34,5	35	40	39,5	42	32
Dejanska hitrost	30,5	32,5	26	31,5	7,5	29	32

Največja odstopanja *Hitrosti* so nastopila v peti *Iteraciji*, ko sta se *Razvojni skupini* pridružila dva nova člana, ter v šesti *Iteraciji,* ko je skupina v želji da bi dosegla prvotno določen rok, preveč optimistično ocenila načrtovano *Hitrost*. Pri *Iteracijah*, v katerih je bila dejanska *Hitrost* bistveno nižja od planirane (npr. v tretji *Iteraciji* je bila dejanska *Hitrost* 26 točk, kar je za 9 točk manj od planiranih 35 točk) je bila v naslednji *Iteraciji* povečana načrtovana *Hitrost,* da bi »ujeli« povprečno načrtovano *Hitrost* (npr. v četrti *Iteraciji* je bila načrtovana *Hitrost* 40 točk). Vendar pa dejanska *Hitrost Razvojne skupine* ni nikoli presegla 32,5 točk. Z načrtovanjem povečevanja *Hitrosti* cilj ni bil dosežen. Možna rešitev je, da bi že v tretji *Iteraciji* preuredili *Seznam Nalog Iteracije* ter *Seznam Uporabniških zgodb*, kar bi omogočilo doseganje roka, četudi z manjšim številom potrjenih *Uporabniških zgodb*. Če ne prej, potem bi to bilo potrebno storiti v peti *Iteraciji*, ki je bila najmanj učinkovita.

Povprečna načrtovana *Hitrost* je bila 36,5 točk, povprečna dejanska *Hitrost* pa 27 točk upoštevajoč vse *Iteracije* in 30,25 točk brez upoštevanja pete *Iteracije.* Pri naslednji *Izdaji* bi ob nespremenjeni sestavi *Razvojne skupine* bilo smiselno upoštevati *Hitrost* 30,25 točk.

Sprotno spremljanje *Hitrosti* omogoča realno načrtovanje *Iteracije*, saj s prilagajanjem načrtovane *Hitrosti* dejanski *Hitrosti* lahko bolj realno ocenimo kapaciteto *Razvojne skupine* ter število *Uporabniških zgodb*, ki jih je možno uresničiti do zastavljenega roka *Izdaje*.

Pri večini *Iteracij* je bila dejanska *Hitrost* nižja od načrtovane. *Razvojna skupina* je navedla naslednje razloge:

- slabša izvedba *Načrtovalnega sestanka Iteracije*;
- spremembe v *Razvojni skupini*;
- preveč optimistično načrtovanje;
- manjša skrb za skladnost z metodologijo *Scrum* po določenem času;
- manjši nesporazumi z *Produktnim vodjo* ipd.

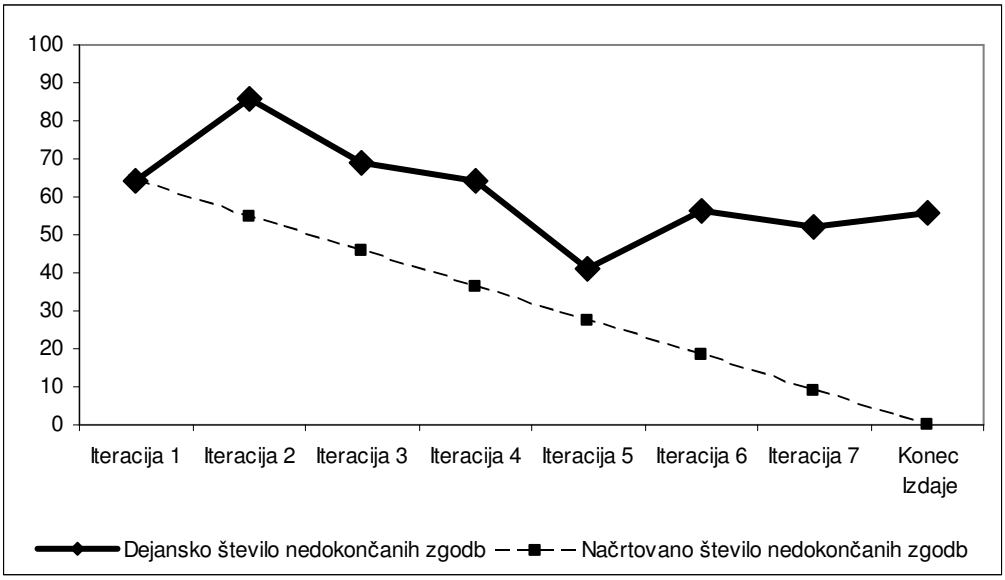
Razvojna skupina je opazila, da je *Hitrost* tesno povezana s kakovostjo načrtovanja *Iteracije* ter da so bile *Iteracije*, ki so potekale slabše, tudi na samem začetku manj natančno načrtovane.

5.3.2 Spremljanje napredka Izdaje

Pri spremljanju napredka *Izdaje* opazujemo število nepotrjenih *Uporabniških zgodb* po vsaki *Iteraciji*, za razliko od spremljanja *Hitrosti*, pri kateri opazujemo število potrjenih *Uporabniških zgodb* po vsaki *Iteraciji*.

Na Sliki 23 je prikazan *Padajoči diagram spremljanja napredka Izdaje*.

Slika 23:Padajoči diagram spremljanja napredka Izdaje



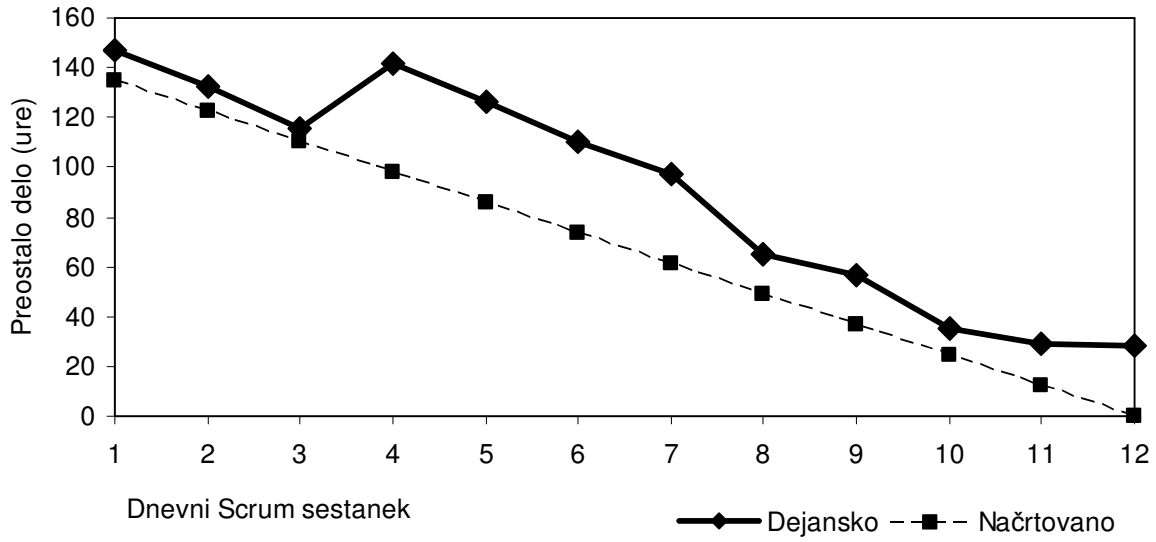
Iz Slike 28 je razvidno, da se je število nepotrjenih *Uporabniških zgodb* v vsaki *Iteraciji* povečevalo, namesto da bi se zmanjševalo. Iz Slike 27, ki prikazuje spremljanje *Hitrosti* pa je razvidno, da je bilo veliko število *Uporabniških zgodb* potrjenih. Razlog je bil v dodajanju novih *Uporabniških zgodb* s strani *Produktnega vodje*, kar je vplivalo na zamik roka *Izdaje*. Dejansko število nepotrjenih *Uporabniških zgodb* se je najbolj približalo načrtovanemu v peti *Iteraciji*. Trend je izračunan za sedem *Iteracij,* kolikor jih je bilo prvotno načrtovanih. Če bi v peti *Iteraciji* ponovno izvedli oceno prioritet *Uporabniških zgodb* in preuredili *Seznam Uporabniških zgodb*, potem bi obstajala možnost, da bi zaključili *Izdajo* v roku. Podobno smo ugotovili tudi ob analizi sprememb *Hitrosti*.

5.4 Spremljanje učinkovitosti prve *Iteracije*

Spremljanje poteka *Iteracije* je podobno spremljanju poteka *Izdaje*, vendar prikazuje število preostalih ur, potrebnih za dokončanje *Nalog* po zaključku *Dnevnega Scrum sestanka*. *Padajoči diagram spremljanja napredka Iteracije* se posodablja vsak dan.

Iz Slike 24 je razvidno, da je v prvi *Iteraciji* dejanski znesek preostalega časa, potrebnega za dokončanje *Nalog*, bil vedno večji od načrtovanega, kar pomeni, da je bila *Razvojna skupina* stalno v zamudi. Najbolj se je približala ciljni vrednosti na tretjem, osmem in desetem *Dnevnem Scrum sestanku.*

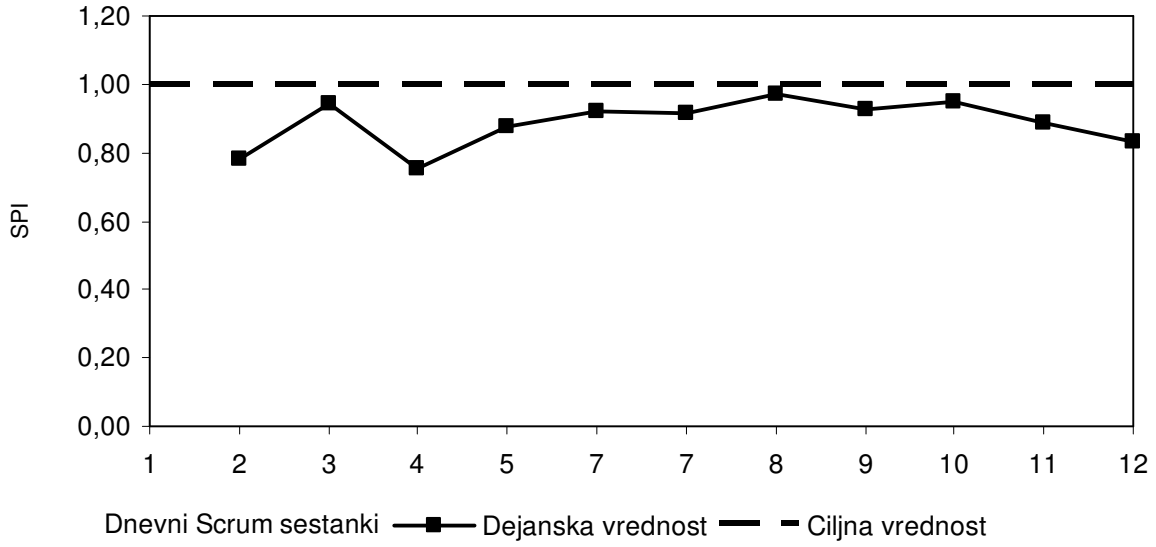
Slika 24: Padajoči diagram spremljanja napredka prve Iteracije



Prvi večji zaostanek se je zgodil na četrtem *Dnevnem Scrum sestanku*. Če bi takrat ukrepali z preureditvijo *Seznama Nalog Iteracije*, bi še lahko dosegli cilj in sicer, da je znesek preostalega časa ob zaključku zadnjega dne enak 0.

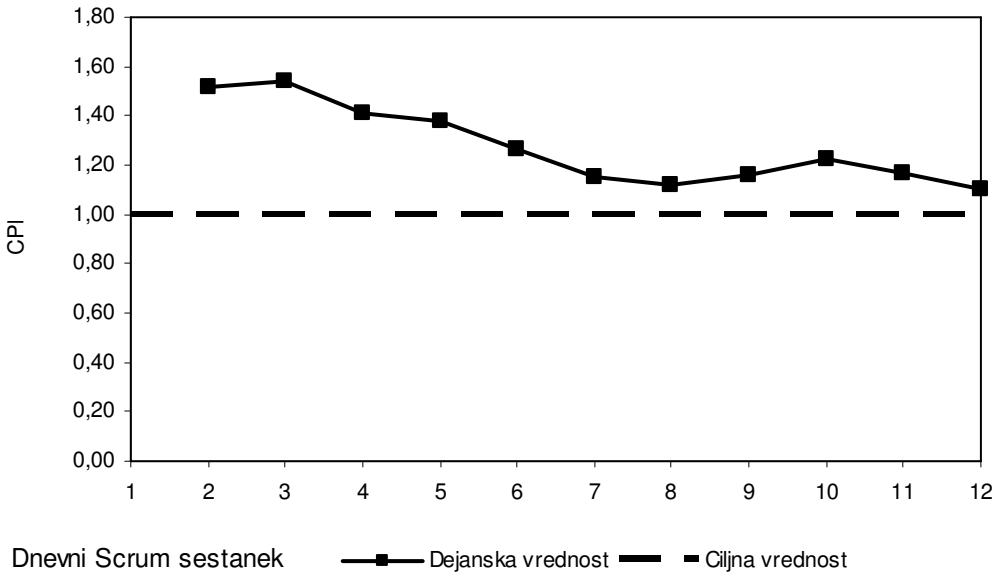
Iz Slike 25 je razvidno, da je dejanski znesek preostalega časa, potrebnega za dokončanje *Nalog*, bil vedno večji od načrtovanega, kar pomeni, da je bila *Razvojna skupina* stalno v zamudi. Najbolj se je približala ciljni vrednosti na tretjem, osmem in desetem *Dnevnem Scrum sestanku*. Ugotovitve so skladne z rezultati analize *Padajočega diagrama napredka Iteracije.*

Slika 25: SPI diagram za prvo Iteracijo



Iz Slike 26 je razvidno, da je *Razvojna skupina* bila ves čas prve *Iteracije* stroškovno učinkovita, kar pomeni, da je porabila manj ur kot je bilo načrtovano.

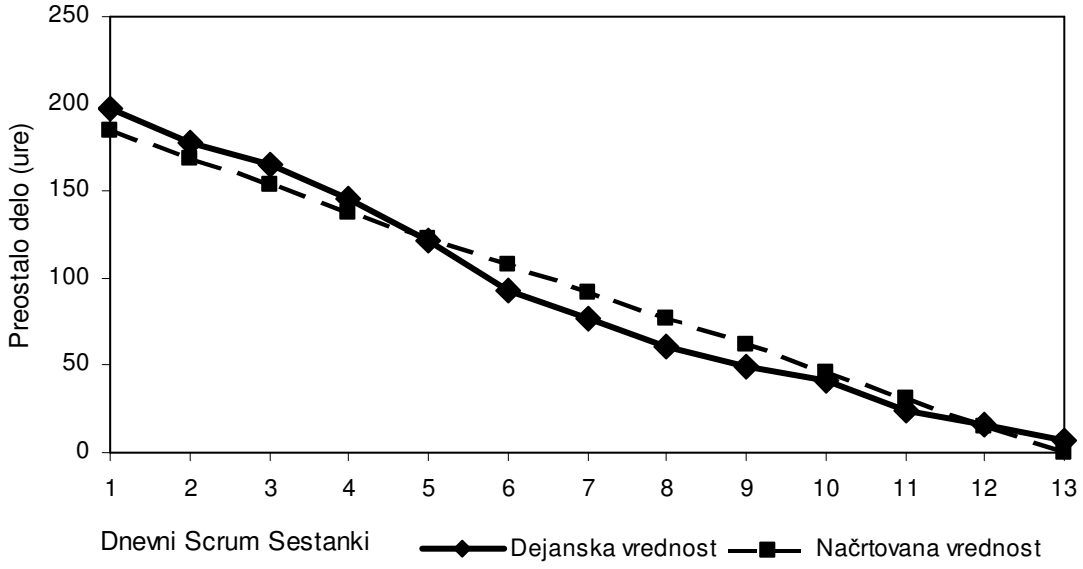
Slika 26: CPI diagram za prvo Iteracijo



5.5 Spremljanje učinkovitosti druge *Iteracije*

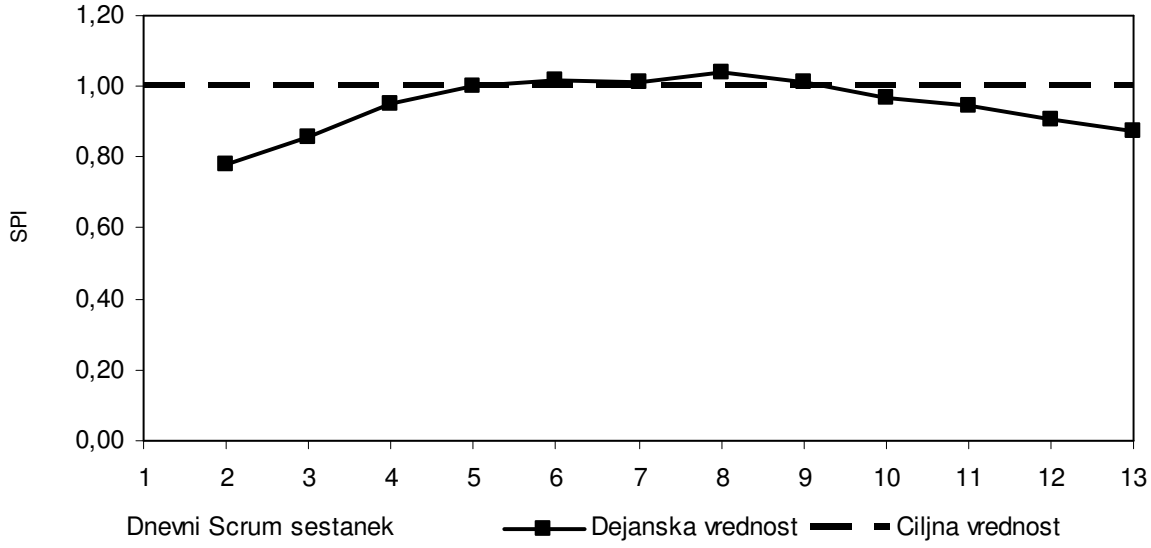
Iz Slike 27 je razvidno, da je bil dejanski znesek preostalega časa, potrebnega za dokončanje *Nalog* v drugi *Iteraciji* večinoma manjši od načrtovanega, kar pomeni, da je bila *Razvojna skupina* večino časa pred rokom. Lahko bi sklepali, da so ocene preostalega časa bile pesimistične, ter da bi *Razvojna skupina* lahko napredovalo z večjo *Hitrostjo.*

Slika 27: Padajoči diagram spremljanja napredka druge Iteracije



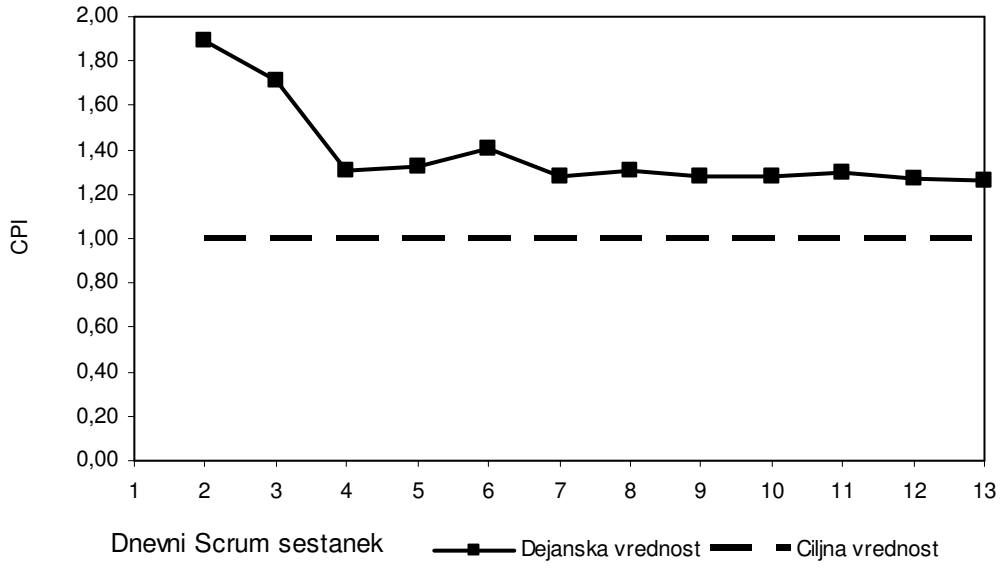
Iz Slike 28 je razvidno, da je SPI diagram za drugo *Iteracijo* skladen s *Padajočim diagramom spremljanja napredka druge Iteracije*.

Slika 28: SPI diagram za drugo Iteracijo



Iz Slike 29 je razvidno, da je bila *Razvojna skupina* zelo učinkovita, saj je ves čas *Iteracije* porabila manj ur, kot je bilo načrtovano. Lahko bi sklepali, da so bile ocene zahtevnosti *Nalog* previsoke.

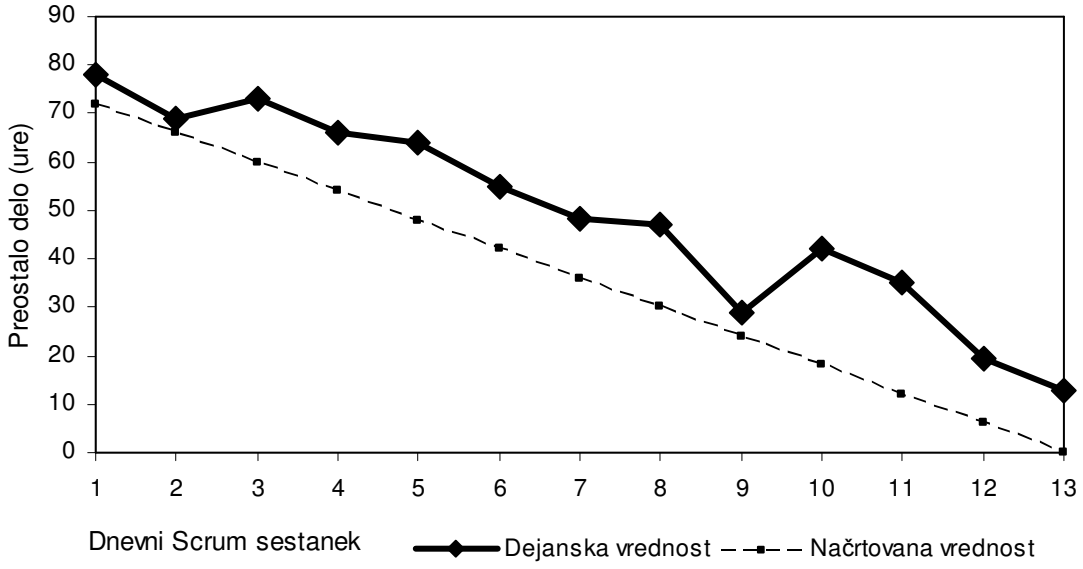
Slika 29: CPI diagram za drugo Iteracijo



5.6 Spremljanje učinkovitosti tretje *Iteracije*

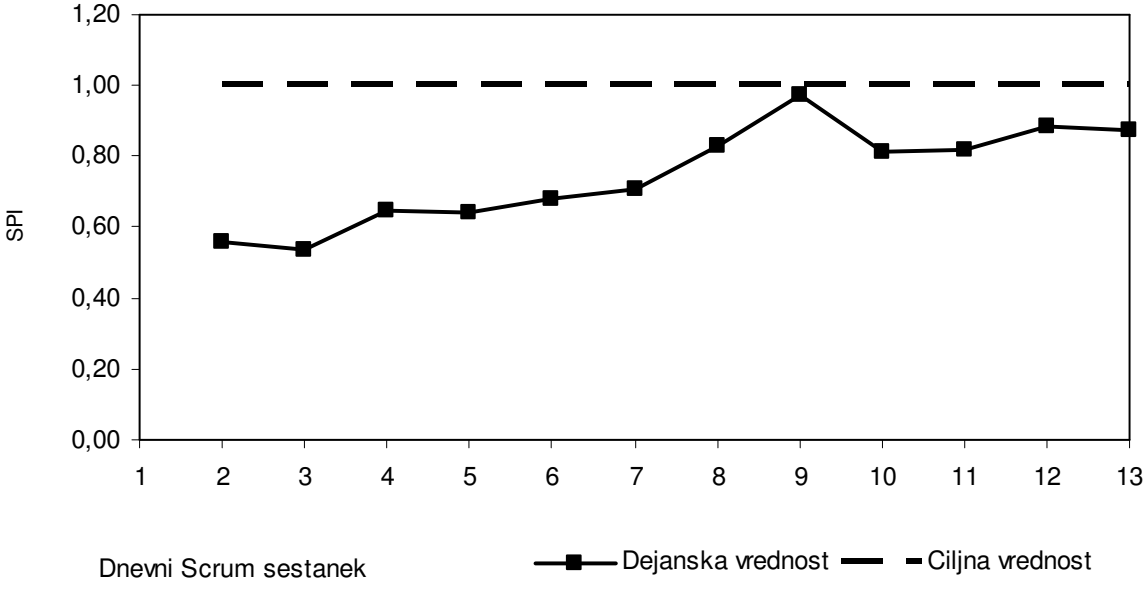
Iz Slike 30 je razvidno, da v tretji *Iteraciji Razvojna skupina* večino časa ni dosegala ciljnih vrednosti in ni uspela dokončati vseh *Nalog* v roku. Najbolj se je približala ciljnim vrednostim v drugem in devetem dnevu. Če bi ukrepali tretji dan *Iteracije*, bi lahko preprečili nedokončane *Naloge* ob zaključku *Iteracije.*

Slika 30: Padajoči diagram spremljanja napredka tretje Iteracije



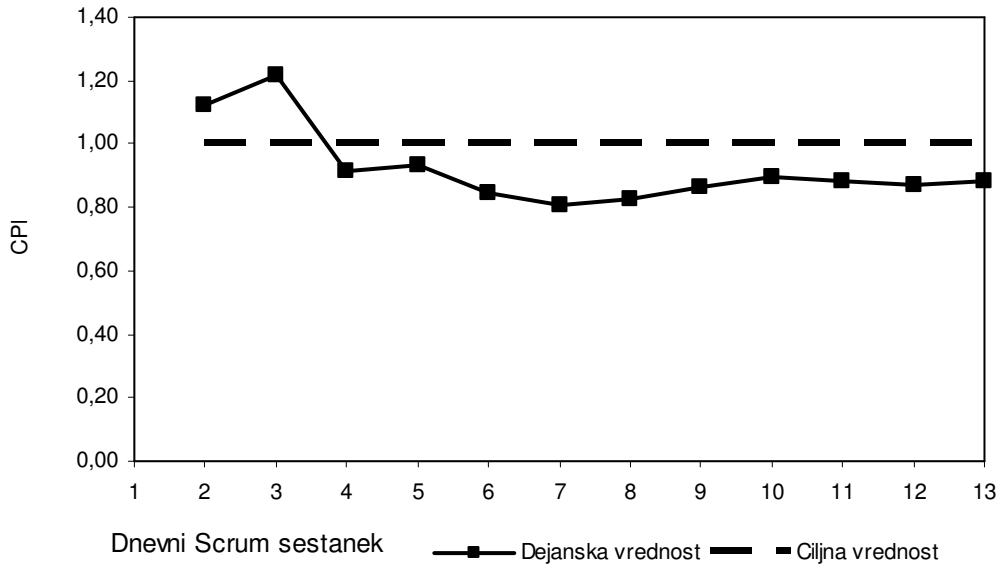
Iz Slike 31 je razvidno, da je SPI diagram za tretjo *Iteracijo* skladen s *Padajočim diagramom spremljanja napredka tretje Iteracije.*

Slika 31: SPI diagram za tretjo Iteracijo



Iz Slike 32 je razvidno, da je bila *Razvojna skupina* stroškovno učinkovita drugi in tretji dan, preostale dneve pa je porabila več ur, kot je bilo načrtovano.

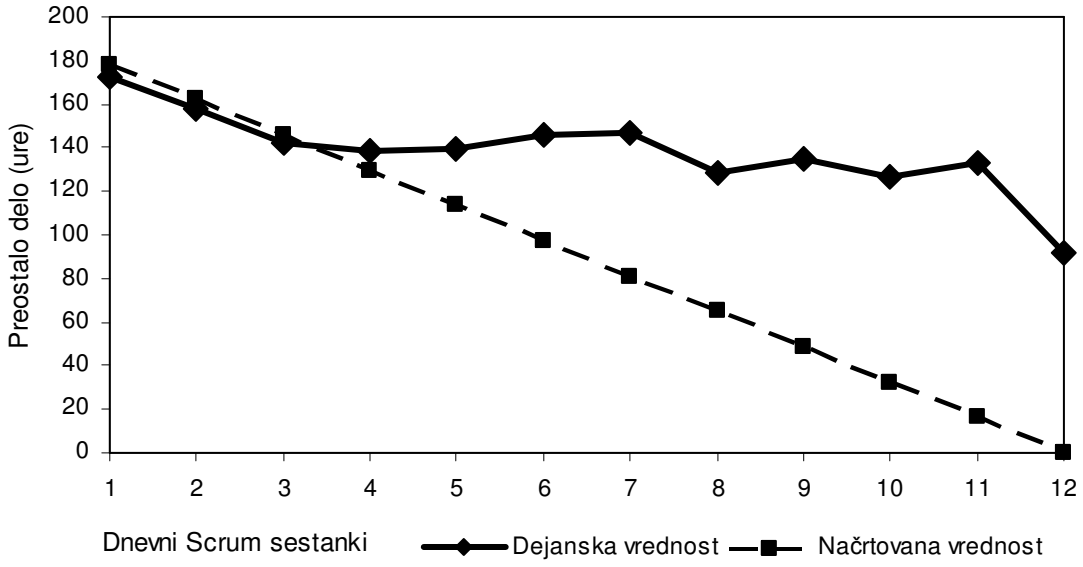
Slika 32: CPI diagram za tretjo Iteracijo



5.7 Spremljanje učinkovitosti četrte *Iteracije*

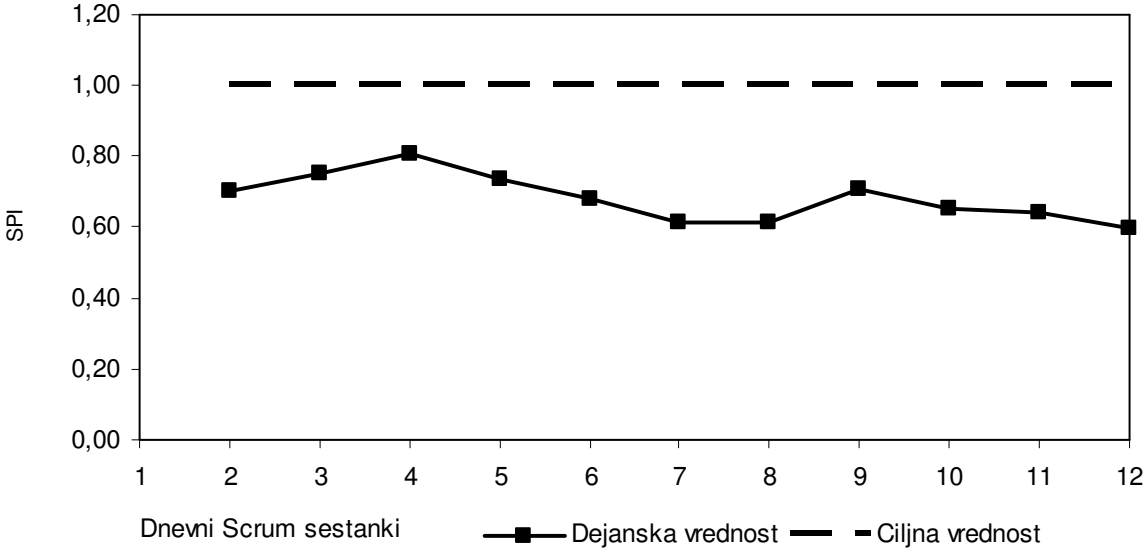
Iz Slike 33 je razvidno, da je *Razvojna skupina* bilo v zaostanku celotno četrto *Iteracijo*. Zaostanek se je po četrtem dnevu začel povečevati in se potem ni znižal do konca *Iteracije*. Če bi ukrepali po četrtem dnevu, bi lahko zaostanek preprečili.

Slika 33: Padajoči diagram spremljanja napredka četrte Iteracije



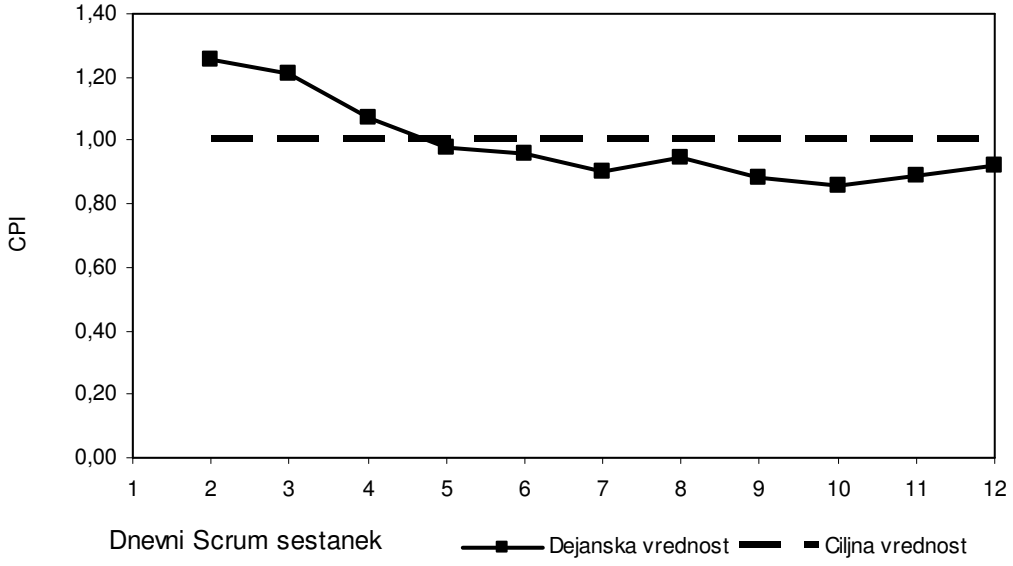
Iz Slike 34 je razvidno, da je SPI diagram za četrto *Iteracijo* skladen s *Padajočim diagram napredka Iteracije.*

Slika 34: SPI diagram za četrto Iteracijo



Iz Slike 35 je razvidno, da čeprav *Padajoči diagram spremljanja napredka Iteracije* in SPI diagram za četrto *Iteracijo* kažeta, da je bila *Razvojna skupina* ves čas v zamudi, je bila istočasno stroškovno učinkovita, kar pomeni, da je porabila enako ali manj ur, kot je bilo ocenjeno glede na zahtevnost posamezne *Naloge*.

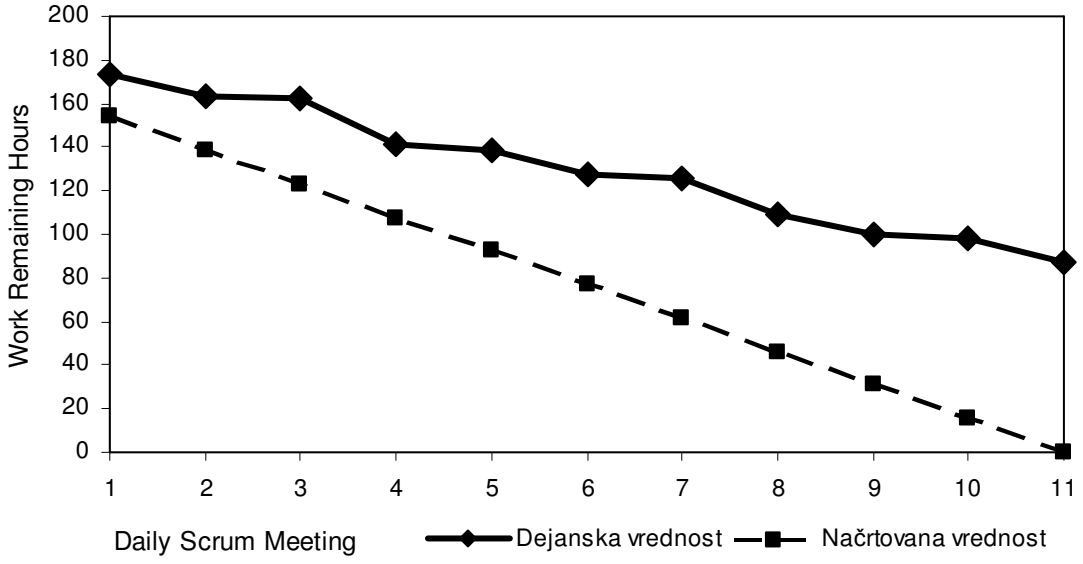
Slika 35: CPI diagram za četrto Iteracijo



5.8 Spremljanje učinkovitosti pete *Iteracije*

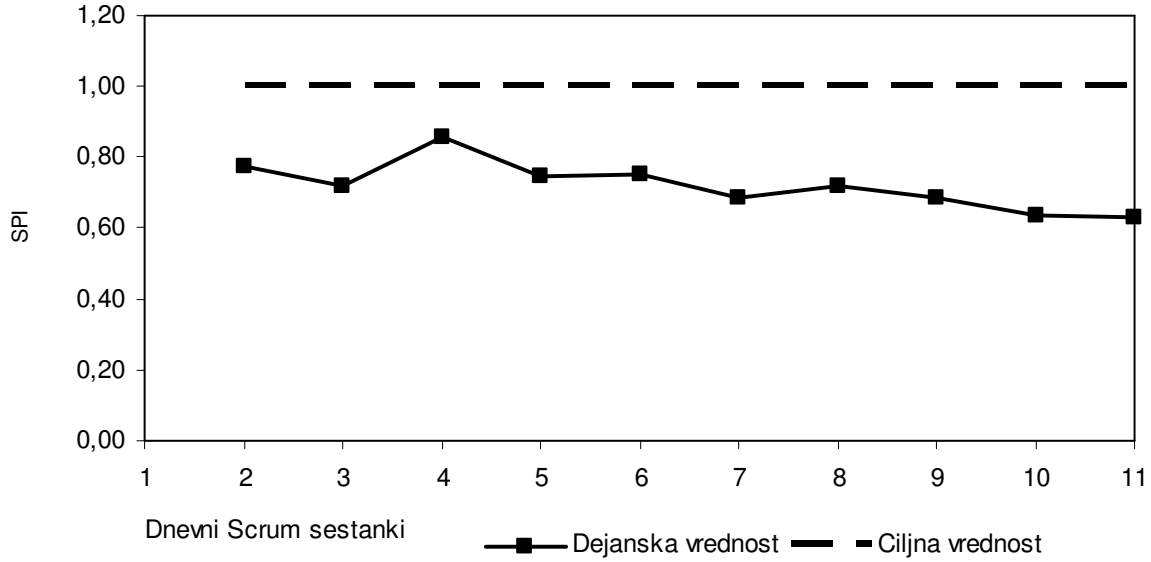
Iz Slike 36 je razvidno, da je *Razvojna skupina* tudi v peti *Iteraciji* zamujala, vendar je zamuda postala večja po drugem dnevu in se do konca *Iteracije* ni zmanjšala. Če bi ukrepali po drugem dnevu, bi lahko zaostanek preprečili ali vsaj zmanjšali. V tej *Iteraciji* se je zgodila sprememba članov *Razvojne skupine*, kar je vplivalo na povečanje obsega dela pri prenosu znanja novim članom in posledično na zmanjšanje obsega dela na razvoju.

Slika 36: Padajoči diagram spremljanja napredka pete Iteracije



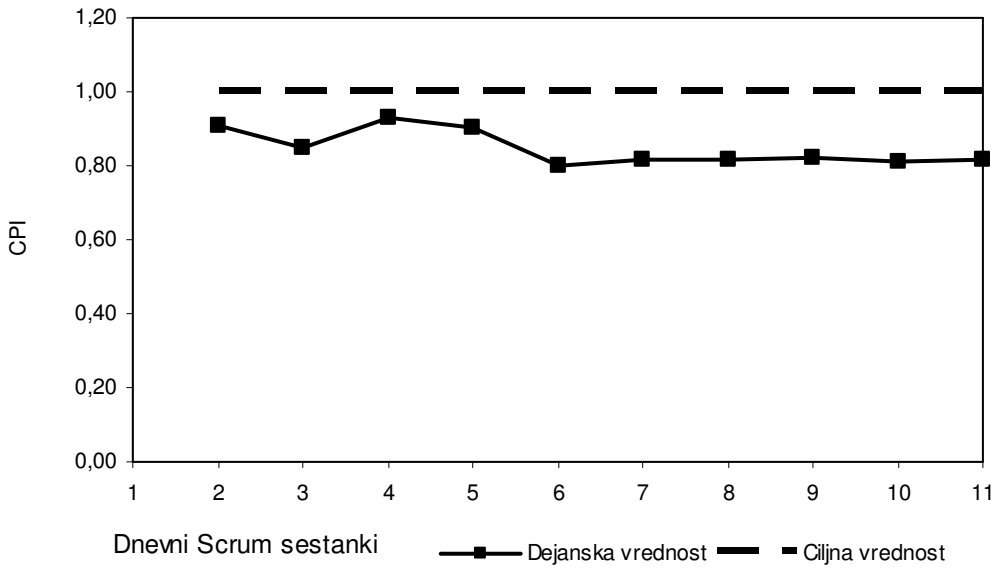
Iz Slike 37 je razvidno, da je SPI diagram za peto *Iteracijo* skladen z *Padajočim diagram napredka pete Iteracije*.

Slika 37: SPI diagram za peto Iteracijo



Iz Slike 38 je razvidno, da *Razvojna skupina* ni bila stroškovno učinkovita in je porabila več ur, kot je bilo ocenjeno glede na zahtevnost posamezne *Naloge*.

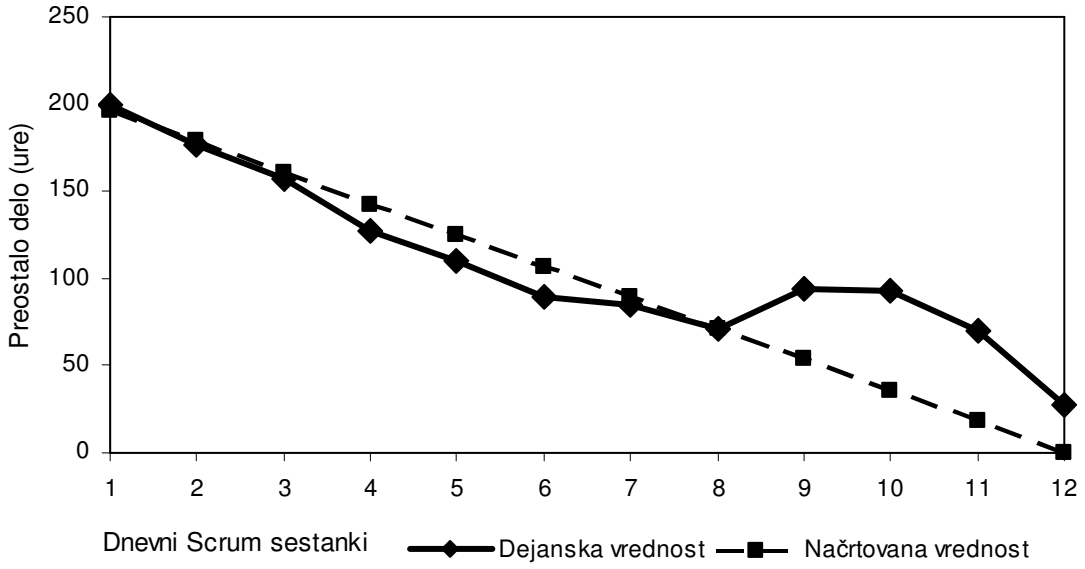
Slika 38: CPI diagram za peto Iteracijo



5.9 Spremljanje učinkovitosti šeste *Iteracije*

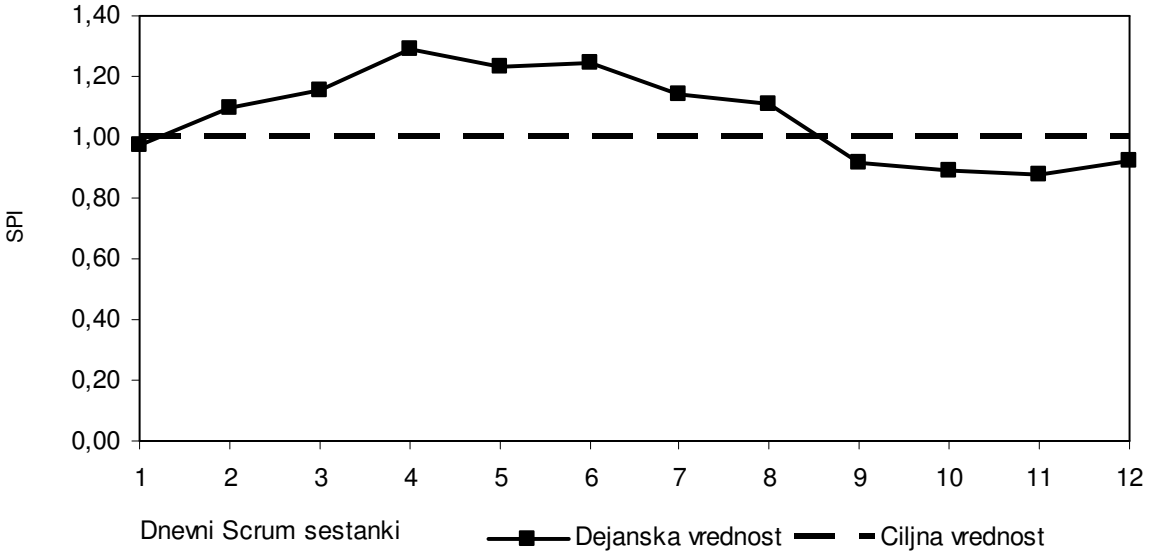
Iz Slike 39 je razvidno, da je *Razvojna skupina* dosegala roke do osmega dneva, potem je deveti, deseti in enajsti dan imela precej zamude, ki jo je nekoliko zmanjšala v dvanajstem dnevu. Če bi ukrepali deveti dan, bi lahko zaostanek preprečili ali vsaj zmanjšali.

Slika 39: Padajoči diagram spremljanja napredka šeste Iteracije



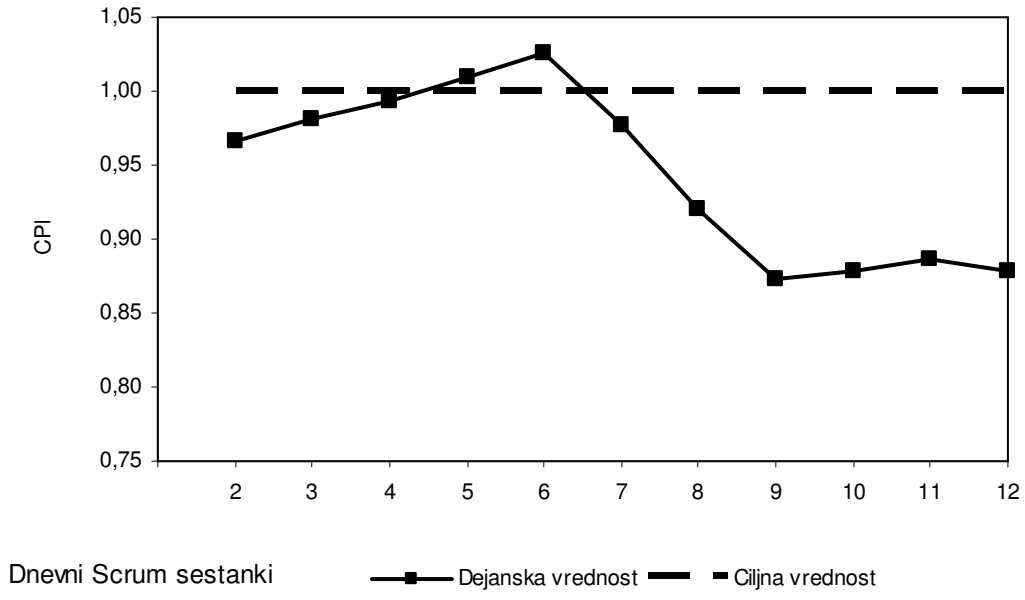
Iz Slike 40 je razvidno, da je SPI diagram za šesto *Iteracijo* skladen s *Padajočim diagram napredka šeste Iteracije.*

Slika 40: SPI diagram za šesto Iteracijo



Iz Slike 41 je razvidno, da *Razvojna skupina* ni bila stroškovno učinkovita celotno *Iteracijo*, z izjemo petega in šestega dne, izrazito neučinkovita pa je bila po osmem dnevu.

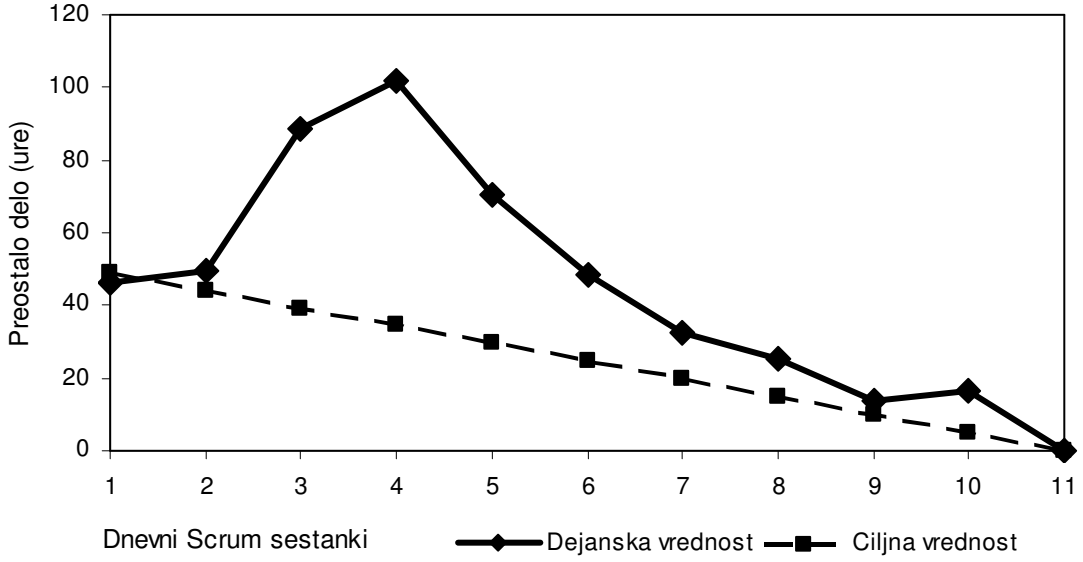
Slika 41: CPI diagram za šesto Iteracijo



5.10 Spremljanje učinkovitosti sedme *Iteracije*

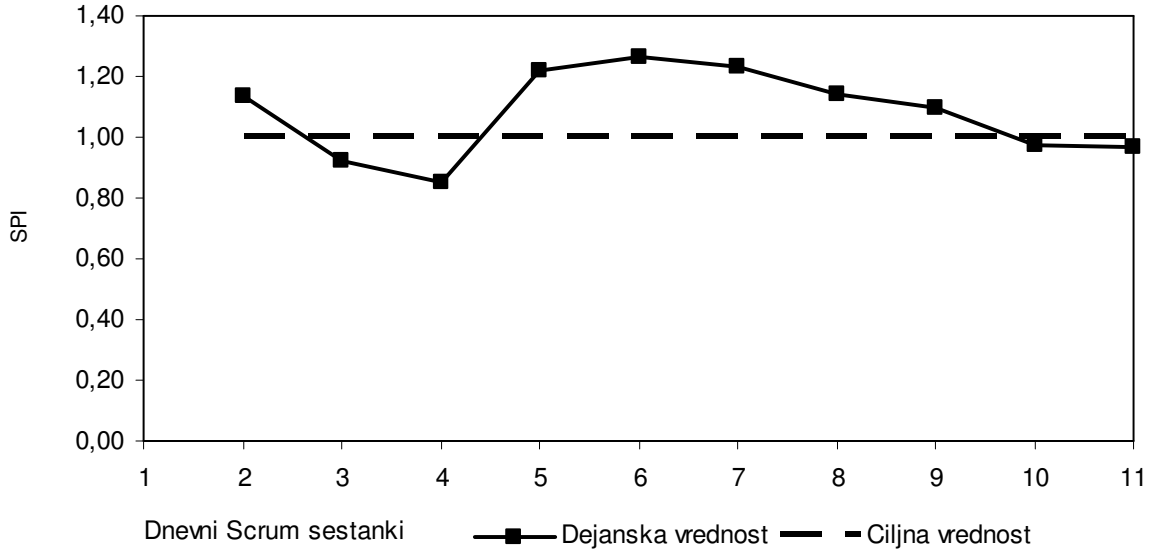
Iz Slike 42 je razvidno, da je *Razvojna skupina* dosegala roke prvi in drugi dan, potem je izrazito zamujala do devetega dne, ko se je zamuda začela zmanjševati in se je zadnji dan zmanjšala na nič.

Slika 42: Padajoči diagram spremljanja napredka sedme Iteracije



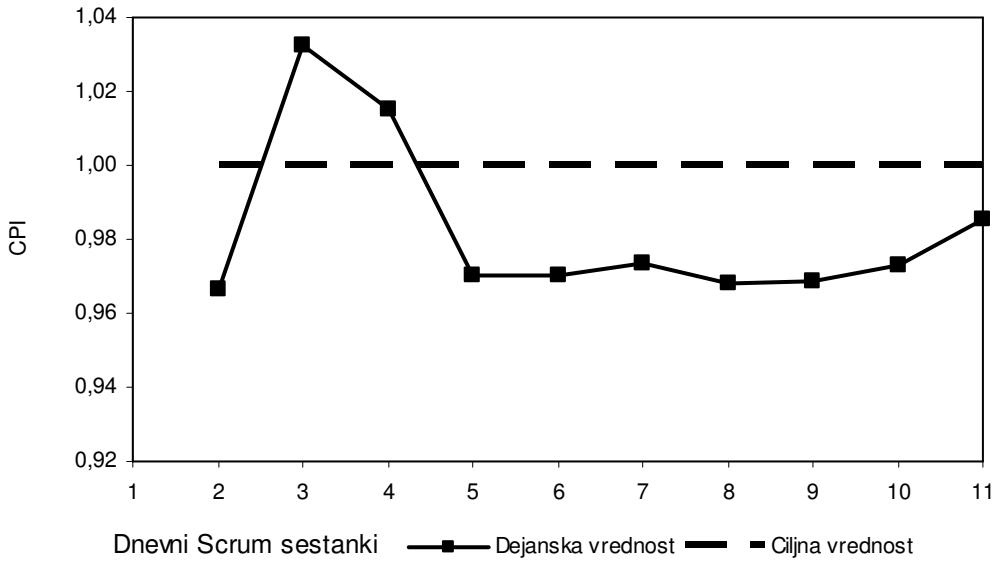
Iz Slike 43 je razvidno, da je SPI diagram za sedmo *Iteracijo* skladen s *Padajočim diagram napredka Iteracije.*

Slika 43: SPI diagram za sedmo Iteracijo



Iz Slike 44 je razvidno, da je bila *Razvojna skupina* stroškovno učinkovita samo tretji in četrti dan.

Slika 44: CPI diagram za sedmo Iteracijo



5.11 Točnost ocenjevanja

Ocena zahtevnosti *Uporabniških zgodb* ter ocena preostalega časa za izvedbo naloge sta ključnega pomena za načrtovanje. V praksi se je izkazalo, da so ocene pogosto preveč optimistične, kar ima za posledico zamude. S namenom doseganja realnih ocen je bila pri projektu uporabljena metoda *Načrtovalni Poker*. Pri ocenjevanju je sodelovalo pet članov *Razvojne skupine* in *Skrbnik Metodologije Scrum*.

Za izračun točnosti ocen so bili uporabljeni podatki o ocenjenem in dejansko porabljenem času. Izvedena je bila primerjava med začetno oceno zahtevnosti *Uporabniških zgodb* in dejansko zahtevnostjo *Uporabniških zgodb*. Pri tem je bilo upoštevano, da ena točka pomeni šest ur. Analiza je bila izvedena na 49 izbranih zgodbah od skupaj 100 zgodb, pri katerih je kakovost zbranih podatkov o dejansko porabljenem času bila ustrezna, saj v zadnjih dveh *Iteracijah* podatki o porabljenem času niso bili evidentirani.

V nadaljevanju je prikazana točnost ocen za posamezno *Iteracijo*, za vse *Uporabniške zgodbe*, ki so bile ocenjene v tej *Iteraciji*, ne glede na to, ali so bile uvrščene v *Seznam Nalog tekoče Iteracije*. Če je katera izmed *Uporabniških zgodb* bila ocenjena večkrat, potem je bila upoštevana zadnja ocena. Zaradi navedenega pristopa število točk za posamezno *Iteracijo* ni primerljivo s podatki o hitrosti za posamezno *Iteracijo*, pri kateri so upoštevane samo tiste *Uporabniške zgodbe*, ki so bile uvrščene v *Seznam Nalog tekoče Iteracije*.

Točnost ocen je prikazana za *Iteracije* 3, 4, 5, 6 in 7. Za prvi dve *Iteraciji* točnost ocen ni prikazana, zaradi pomanjkljive kakovosti podatkov.

V Tabeli 44 so navedeni podatki o točnosti ocen za izbrane *Iteracije* glede na povprečje ocen. Iz Tabele 44 je razvidno, da so bile začetne ocene preveč optimistične, saj je bila povprečna dejanska zahtevnost *Uporabniških zgodb* enaka 4,05 in je bila večja od načrtovane zahtevansti, ki je bila izračunana kot povprečje ocen in je bila enaka 3,47. Kazalnik BRE je enak 0,65 in je večji od ciljne vrednosti 0.

Tabela 44: Točnost ocen Uporabniških zgodb (Povprečje)

Iteracija	AvgE	A	AE avg	RE avg	REbias avg	BRE avg	BREbias avg
3	3,27	4,28	2,43	0,55	-0,12	0,93	0,27
4	3,35	2,69	1,40	0,67	-0,51	0,72	-0,46
5	4,66	6,20	2,70	0,34	0,11	0,57	0,35
6	2,94	2,93	1,04	0,39	-0,08	0,51	0,06
7	3,37	5,00	1,72	0,29	0,21	0,51	0,42
Skupaj	**3,47**	**4,05**	**1,81**	**0,45**	**-0,10**	**0,65**	**0,11**

V Tabeli 45 so navedeni podatki o točnosti ocen za izbrane *Iteracije* glede na povprečje ocen. Iz Tabele 45 je razvidno, da so bile začetne ocene preveč optimistične, saj je bila povprečna dejanska zahtevnost *Uporabniških zgodb* enaka 4,05 in je bila večja od načrtovane zahtevnosti, ocenjene z uporabo *Načrtovalnega pokra*, ki je enaka 3,13. Kazalnik BRE je enak 0,50 in je večji od ciljne vrednosti 0.

Tabela 45: Točnost ocen Uporabniških zgodb (Načrtovalni poker)

Iteracija	ppE	A	AE pp	RE pp	REbias pp	BRE pp	BREbias pp
3	2,59	4,28	2,04	0,33	0,13	0,65	0,45
4	2,50	2,69	1,06	0,40	-0,11	0,51	0,01
5	4,80	6,20	1,80	0,23	0,18	0,42	0,37
6	2,69	2,93	1,09	0,32	-0,01	0,48	0,17
7	3,50	5,00	1,50	0,24	0,24	0,37	0,37
Skupaj	**3,13**	**4,05**	**1,48**	**0,31**	**0,06**	**0,50**	**0,26**

6. ZAKLJUČEK

6.1 Povzetek doktorske disertacije

Rezultat doktorske disertacije je model *AGIT* za spremljanje učinkovitosti agilnega razvoja programske opreme, ki temelji na sprotnem spremljanju ključnih kazalnikov in združuje spoznanja s področja agilnih metod razvoja programske opreme, področja merjenja učinkovitosti razvojnih procesov, področja podatkovnih skladišč in področja poslovne inteligence.

Doktorska disertacija vsebuje pregled opis razvoja programske opreme in projektnega vodenja ter agilnega razvoja programske opreme in predstavitev problematike spremljanja učinkovitosti agilnega razvoja programske opreme ter rezultate dosedanjih raziskav na obravnavanem področju.

Model *AGIT* je možno uporabiti kot osnovo za razlago in razumevanje procesa spremljanje učinkovitosti agilnega razvoja programske opreme, njegova uporaba pa prispeva k povečanju učinkovitosti agilnega razvoja programske opreme.

6.2 Pregled prispevkov znanosti

Ključni prispevek doktorske disertacije k znanosti je izdelava celovitega modela *AGIT* za spremljanje učinkovitosti agilnega razvoja programske opreme. V okviru modela za spremljanje učinkovitosti agilnega razvoja programske opreme so prispevki k znanosti:

- opredelitev ključnih kazalnikov za spremljanje učinkovitosti;
- opredelitev metrik, potrebnih za izračun ključnih kazalnikov;
- opredelitev postopkov za zbiranje podatkov;
- opredelitev organizacije podatkov in
- opredelitev oblike poročanja.

Model *AGIT* smo razvili po korakih, ki sledijo konceptu PPS. Najprej smo opisali razvojni proces razvoja programske opreme z uporabo metodologije *Scrum*. Potem smo določili deležnike razvojnega procesa: Poslovodstvo, *Produktni vodja*, *Skrbnik Metodologije Scrum* in *Razvojna skupina*. Za vsakega izmed deležnikov smo določili cilje, za vsakega izmed ciljev pa kazalnike učinkovitosti. Nato smo podali podroben opis kazalnikov učinkovitosti za posameznega deležnika. Sledila je opredelitev ciljnih in mejnih vrednosti kazalnikov učinkovitosti. Potem smo določili način zbiranja podatkov, pri čemer smo sledili agilnim načelom ter oblikovali podatkovni model. Na koncu smo določili tudi način prikazovanja podatkov.

Uporabnost modela *AGIT* smo preverili na študijskem primeru. V doktorski disertaciji smo najprej predstavili podjetje in projekt ter podmnožico kazalnikov modela *AGIT*, ki smo jo uporabili pri preverjanju modela: *Diagram Hitrosti, Padajoči diagram spremljanja napredka Izdaje, Padajoči diagram spremljanja napredka Iteracije, SPI diagram, CPI diagram* in *BRE tabela*. Primer smo opisali iz časovnega, organizacijskega in kakovostnega vidika. Predstavili smo potek *Izdaje* in *Iteracije* ter rezultate projekta. Sledila je predstavitev rezultatov merjenja učinkovitosti na ravni *Izdaje* ter na ravni posamezne *Iteracije*, za prvih sedem *Iteracij*. Podali

smo tudi rezultate analize točnosti ocen zahtevnosti *Uporabniških zgodb*. Ob preverbi smo ugotovili, da je model AGIT skladen z načeli agilnosti glede načina zbiranja in prikazovanja podatkov.

Navedeno vsebuje povzetek dobre prakse na področju spremljanja kazalnikov ter področju poslovne inteligence.

6.3 Smernice za nadaljnje raziskave

V uvodnem poglavju doktorske disertacije smo navedli predpostavke in omejitve, ki so istočasno tudi smernice za nadaljnje raziskave:

Predpostavka doktorske disertacije je, da je model *AGIT*, čeprav temelji na metodologiji *Scrum,* uporaben tudi pri drugih agilnih metodologijah. V nadaljnjih raziskavah bi bilo smiselno preizkusiti uporabo modela *AGIT* na drugih agilnih metodah.

Potek nadaljnjih raziskav je možen v okviru razvojnih nalog Laboratorija za programsko opremo na Fakulteti za računalništvo in informatiko v Ljubljani:

- nadaljevanje uporabe modela *AGIT* v akademskem okolju;
- nadaljevanje uporabe modela *AGIT* v industrijskem okolju;
- nadaljevanje raziskav na področju planiranja in ocenjevanja;
- nadaljevanje raziskav na področju orodij za zbiranje in prikazovanje podatkov modela *AGIT*.

Ena izmed možnih smernic nadaljnjih raziskav je tudi raziskava možnosti uporaba modela *AGIT* pri revidiranju agilnega razvoja programske opreme.

Pojmovnik

Angleški pojem	Slovenski pojem	Opis
AGIT	*AGIT*	Agilna Informacijska Tehnologija, model spremljanja učinkovitosti agilnega razvoja programske opreme
Burndown Diagram	*Padajoči diagram spremljanja napredka* *»Izgorevalni diagram«*	Padajoči diagram spremljanja napredka
Burnup Diagram	*Rastoči diagram spremljanja napredka* *»Dogorevalni diagram«*	Rastoči diagram spremljanja napredka
Cummulative Flow Diagram	*Kumulativni diagram poteka*	Kumulativni prikaz poteka projekta
Development Team	*Razvojna skupina* *»Razvojno moštvo«*	Skupina razvijalcev, ki ustvarja programsko opremo
Done	*Potrjena*	Status *Uporabniške zgodbe*, ki pomeni, da je odobrena s strani *Produktnega vodje*
Feature	*Funkcionalnost*	Lastnost izdelka
Impediment	*Ovira*	Težava, s katero se je srečala *Razvojno moštvo* pri svojem delu in ki jo rešuje *Skrbnik metodologije*
Line of Balance (LOB)	*Linija ravnotežja*	Vrsta kumulativnega prikaza poteka projekta
Planning Poker	*Načrtovalni poker*	Metoda za ocenjevanje zahtevnosti realizacije *Uporabniških zgodb*
Priority	*Prioriteta Uporabniške zgodbe*	Ocena poslovne prioritete *Uporabniške zgodbe*
Product Backlog	*Seznam Uporabniških zgodb* *Seznam zahtev* *»Zaostanek izdelka«*	Množica uporabniških zgodb v *Izdaji*
Product Owner	*Produktni vodja* *Lastnik izdelka*	Predstavnik uporabnikov, ki je odgovoren za vsebino in prioriteto *Uporabniških zgodb*
Release	*Izdaja*	Izdaja
Release Plan	*Načrt Izdaje*	Časovnica *Izdaje*
Release Planning Meeting	*Sestanek za načrtovanje Izdaje* *Načrtovalni sestanek Izdaje*	Sestanek, katerega rezultat je *Načrt izdaje*
Scrum	*Metodologija Scrum* *Ogrodje Scrum*	Metodologija za projektno vodenje agilnega razvoja programske opreme
ScrumMaster	*Skrbnik metodologije Scrum* *Skrbnik Scrum projekta*	Oseba, ki je odgovorna za nemoten potek Scrum projekta

	»Scrum Mojster«	
Sprint	*Iteracija* *»Tek«* *»Šprint«*	Faza *Izdaje*
Sprint Backlog	*Seznam Nalog Iteracije* *»Zaostanek Šprinta«*	Množica *Nalog*, iz katerih so sestavljene *Uporabniške zgodbe* v *Seznamu Uporabniških zgodb*
Sprint Plan	*Načrt Iteracije*	Časovnica *Iteracije*
Sprint Planning Mmeeting	*Načrtovalni sestanek Iteracije* *Sestanek za načrtovanje Iteracije* *»Načrtovalni sestanek Šprinta«*	Sestanek, katerega rezultat je *Načrt Iteracije*
Sprint Retrospective Meeting	*Retrospektivni sestanek Iteracije* *Sestanek za oceno kakovosti razvojnega procesa* *»Retrospektivni sestanek Šprinta«*	Sestanek, na katerem poteka analiza predhodne *Iteracije,* katerega rezultat so izboljšave razvojnega procesa v naslednji *Iteraciji*
Sprint Review Meeting	*Potrjevalni sestanek Iteracije* *Sestanek za predstavitev rezultatov Iteracije* *»Potrjevalni sestanek Šprinta«*	Sestanek, katerega rezultat je potrditev dokončanih *Uporabniških zgodb* s strani *Produktnega vodje*
Story Point	*Točka*	Mera za oceno zahtevnosti izdelave *Uporabniške zgodbe*
Task	*Naloga*	Naloga, ki je sestavni del *Uporabniške zgodbe* in obsega od 4 do največ 16 ur dela
User Story	*Uporabniška zgodba* *Zahteva*	Način komuniciranja željenih lastnosti izdelka s strani *Lastnika izdelka*
Velocity	*Hitrost*	Število *točk*, ki jih *Razvojna skupina* lahko realizira v eni *Iteraciji*
Work In Progress	*Nedokončane Uporabniške zgodbe*	Nedokončane *Uporabniške zgodbe*
Work Remaining	*Preostali čas*	Ocena časa, ki je potreben za dokončanje *Naloge*
Work Spent	*Porabljeni čas*	Čas, ki je bil porabljen za izdelavo *Naloge*

Literatura

[Abr02] ABRAHAMSSON P., SALO O., RONKAINEN J., WARSTA J.: »Agile software development methods, Review and analysis«. VTT Electronic, Espoo (2002). http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf <na dan: 17.12.2012>

[All03] ALLEMAN G. B., HENDERSON M.: »Making Agile Development Work in a Government Contracting Environment, Measuring velocity with Earned Value«. Proceedings of Agile Development Conference ADC 2003, 25–28 junij, 2003 Salt Lake City, Utah. (2003) http://www.niwotridge.com/PDFs/ADC%20Final.pdf <na dan: 17.12.2012>

[Amb10] AMBLER, S. W.: »The Agile Maturity Model (AMM)«. Dr. Dobbs Journal, april 2010 (2010) http://www.drdobbs.com/architecture-and-design/the-agile-maturity-model-amm/224201005 <na dan: 17.12.2012>

[Amb11] AMBLER S.: »How Successful Are IT Projects, Really?«. Dr. Dobbs, decmeber 2011 (2011) http://www.drdobbs.com/architecture-and-design/how-successful-are-it-projects-really/232300110?pgno=2 <na dan: 17.12.2012>

[ANS98] ANSI: »American National Standards Institute standard on Earned Value Management System Guidelines«, ANSI/EIA-748-A-1998 (R2002) (1998).

[Baj03] BAJEC M., KRISPER M.: »Agilne metodologije razvoja informacijskih sistemov«. Uporabna informatika, april, maj, junij 2003, 11(2), str. 68-76 (2003). http://bajecm.fri.uni-lj.si/CRP2001/Clanki/Agilne%20Metodologije%20Razvoja%20IS.pdf <na dan: 17.12.2012>

[Bar05] BARTON B., SCHWABER K., RAWSTHORNE, D.: »Reporting Scrum Project Progress to Executive Management through Metrics« (January 2005). http://www.danube.com/docs/Scrum_Metrics_for_Management.pdf <na dan: 17.12.2012>

[Bec01a] BECK K., BEEDLE M., VAN BENNEKUM A., COCKBURN A., CUNNINGHAM W., FOWLER M., GRENNING J., HIGHSMITH J., HUNT A., JEFFRIES R., KERN J., MARICK B., MARTIN R., MELLOR S., SCHWABER K., SUTHERLAND J. and THOMAS D.: »Manifest agilnega razvoja programske opreme« (2001). http://agilemanifesto.org/iso/sl/ <na dan: 17.12.2012>

[Bec01b] BECK K., BEEDLE M., VAN BENNEKUM A., COCKBURN A., CUNNINGHAM W., FOWLER M., GRENNING J., HIGHSMITH J., HUNT A., JEFFRIES R., KERN J., MARICK B., MARTIN R., MELLOR S., SCHWABER K., SUTHERLAND J. and THOMAS D.: »Principi v ozadju agilnega manifesta« (2001). http://agilemanifesto.org/iso/sl/principles.html <na dan: 17.12.2012>

[Boe04] BOEHM B., TURNER R.: »Balancing Agility and Discipline – A Guide for the Perplexed«. Pearson Education, ZDA (2004).

[Cab06] CABRI, A., GRIFFITHS, M. (2006): »Earned value and Agile reporting«. Proceedings of AGILE 2006, str. 17 – 22, IEEE Computer Society

Washington, DC, ZDA (2006)

[Car10] CARVER J. C., JACCHERI L., MORASCA S., SHULL F.: » A checklist for integrating student empirical studies with research and teaching goals«. Empirical Software Engineering, 2010. – Vol. 15. – str. 35–59 (2010)

[Ces05] CESCHI, M. Sillitti, A. ; Succi, G. ; De Panfilis, S.: »Project Management in Plan-Based and Agile Companies«, IEEE Software, Vol. 22, Iss: 3, May/June 2005, str. 21-27 (2005).

[Chi04] CHILLARA S., DEEMER P.: »Scrum and CMMI: A High level assessment of compatibility « http://www.cio-index.com/nm/articlefiles/71934-ScrumCMMI.pdf <na dan: 17.12.2012>

[Chr11] CHRISTOPH E. »Earning Value the Agile Way: Using Story Points to Generate EV Metrics«, PMI Earned Value Management Community of Practice: The Measurable News, 2011, Issue 2, str. 17-22 (2011) http://mycpm.org/wp-content/uploads/2012/02/MN-2011.02.pdf#page=17 <na dan: 17.12.2012>

[CMM06] CMMI: »CMMI® for Development (CMMI-DEV), Version 1.2. CMU/SEI-2006-TR-008, Improving processes for better products«. Software Engineering, Institute, Carnegie Mellon University, ZDA (2006). http://www.sei.cmu.edu/reports/06tr008.pdf <na dan: 17.12.2012>

[CMM10] CMMI: »CMMI for Development, Version 1.3, CMMI-DEV, V1.3, CMU/SEI-2010-TR-033, Improving processes for developing better products and services«.
Software Engineering, Institute, Carnegie Mellon University, ZDA (2010). http://www.sei.cmu.edu/reports/10tr033.pdf <na dan: 17.12.2012>

[Coh04] COHN M.: »User stories applied for agile software development«. Boston, MA: Addison-Wesley, ZDA (2004).

[Coh12] COHN M.: »Agile Succeeds Three Times More Often Than Waterfall« (2012) http://www.mountaingoatsoftware.com/blog/agile-succeeds-three-times-more-often-than-waterfall <na dan: 17.12.2012>

[Dam07] DAMIANI E., COLOMBO A., FRATI F., BELLETTINI C.: »A metamodel for modeling and measuring Scrum development process«. XP 2007, LNCS 4536, str. 74-83. (2007)

[DeG90] DeGRACE P., STAHL L.: »Wicked problems, righteous solutions : a catalogue of modern software engineering paradigms«. Englewood Cliffs, N.J. : Yourdon Press, Yourdon Press computing series, ZDA (1990).

[Din08] DINGSØYR T., DYBÅ T., ABRAHAMSSON P.: »A Preliminary Roadmap for Empirical Research on Agile Software Development«. Proceedings of Agile 2008 Conference. 4-8. avgust 2008. Toronto, Canada. Str. 83–94. (2008)

[Din12] DINGSØYR T., NERUR S., BALIJEPALLY V., BREDE MOE N.: »A decade of agile methodologies: Towards explaining agile software development«. Journal of Systems and Software, Volume 85, Issue 6, June 2012, str. 1213-1221. (2012)

[Dru10] DRUCKMAN A.: »Scrum, Rugby and Teamwork«, April 6th, 2010 (2010)

http://blogs.collab.net/agile/scrum-rugby-and-teamwork <na dan: 17.12.2012>

[Dyb08] DYBÅ T., DINGSØYR T.: »Empirical studies of agile software development: A systematic review«, Information and Software Technology, Vol. 50, No. 9-10, 2008, str. 833–859 (2008)

[Eck06] ECKERSON, Wayne, W.: »Performance Dashboards«. John Wiley & Sons, Inc., ZDA. (2006)

[Fen99] FENTON N. E., NEIL M.: »Software metrics: successes, failures, and new directions«. Journal of Systems and Software, (47) 2-3, str. 149-157 (1999) http://www.eecs.qmul.ac.uk/~norman/papers/jss_1999.pdf <na dan: 17.12.2012>

[Har06] HARTMANN D., DYMOND R.: »Appropriate Agile Measurement: Using Metrics and Diagnostics to Deliver Business Value«, Proceedings of AGILE 2006 Conference (AGILE'06), str.126 – 134. IEEE Computer Society Washington, DC, ZDA (2006)

[Hol06] HOLLER R.: »Five Myths of Agile Development«, May 2006, Better Software Magazine (2006) http://www.versionone.com/pdf/AgileMyths_BetterSoftware.pdf <na dan: 17.12.2012>

[Gat10] GAT I.: »Making Agile Work - Standish Group Chaos Reports Revisited«, The Agile Executive, 11.1.2010. (2010) http://theagileexecutive.com/2010/01/11/standish-group-chaos-reports-revisited / <na dan: 17.12.2012>

[Gla08] GLAZER H., DALTON J., ANDERSON D., KONRAD M., SHRUM S.: »CMMI or Agile: Why Not Embrace Both!«, Technical note , CMU/SEI-2008-TN-003. (2008) http://www.sei.cmu.edu/reports/08tn003.pdf <na dan: 17.12.2012>

[Gol04] GOLFARELLI M., RIZZI S., CELLA I.: »Beyond Data Warehousing: What's Next in Business Intelligence?« DOLAP'04, November 12–13, 2004, Washington, DC, ZDA (2004).

[Gre02] GRENNING, J.: »Planning poker or how to avoid analysis paralysis while release planning«. (2002) http://www.renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf <na dan: 17.12.2012>

[Gri10] GRIFFITHS M.: »Lessons Learned in Project Metrics: Are Your Metrics Dumb or Smart?«, February 16th, 2010. (2010) http://leadinganswers.typepad.com/leading_answers/2010/02/smart-metrics-slides.html <na dan: 17.12.2012>

[GSA12] GSA: General Services Administration Acquisition Manual, 534.201 Policy, Amendment 2011– 02, 15. junij, 2011. (2011) https://www.acquisition.gov/gsam/current/html/Part534.html <na dan: 17.12.2012>

[Gup08] GUPTA, S.: »SOX Compliant Agile Processes«, Proceedings of AGILE 2008 Conference (AGILE'08), str. 140-143. IEEE Computer Society Washington, DC, ZDA (2008)

[Har06] HARTMANN D., R. DYMOND R., "Appropriate Agile Measurement:

Using Metrics and Diagnostics to Deliver Business Value", Proceedings of AGILE 2006 Conference (AGILE'06), Minneapolis, Minnesota, 2006, str. 126–134. (2006)

[HMG11] HM Government: »Government ICT Strategy – Strategic Implementation Plan«, October 2011. (2011) http://www.cabinetoffice.gov.uk/content/government-ict-strategy-strategic-implementation-plan <na dan: 17.12.2012>

[Hun09a] HUNDERMARK P.: »Measuring for Results: Metrics and Myths«, predstavitev, 1.9.2009 (2009) http://www.scrumsense.com/wp-content/uploads/2009/10/Measuring-for-Results-2-small.pdf <na dan: 17.12.2012>

[IIA09] IIA: Global Technology Audit Guide (GTAG) 12: Auditing IT Projects. Altamonte Springs: The Institute of Internal Auditors (IIA). (2009)

[IEE04] IEEE: »SWEBOK - Guide to the software engineering body of knowledge : 2004 version«, IEEE Computer Society, Los Alamitos, ZDA (2004)

[IEE08] IEEE: »SWEBOK - Guide to the software engineering body of knowledge«, Chapter 12: »Software Measurement«, Draft, February 4, 2008, IEEE Computer Society, Los Alamitos, ZDA (2008)

[ISA11] ISACA: »Top Business/Technology Issues Survey Results«. Rolling Meadows: Information Systems Audit and Control Association (ISACA). (2011)

[ISO95] International Organization for Standardization (ISO): »ISO/IEC 12207:1995; Information technology – Software life cycle processes« z amandmajema ISO/IEC 12207:1995/Amd 1:2002 in ISO/IEC 12207:1995/Amd 2:2004 (2004).

[ISO04] International Organization for Standardization (ISO): »ISO/IEC 90003:2004; Software engineering – Guidelines for the application of ISO 9001:2000 to computer sotfware« (2004).

[ISO08] International Organization for Standardization (ISO): »ISO/IEC 12207:2008; Systems and Software Engineering – Software life cycle processes« (2008).

[ITG07a] ITGI: CobiT 4.1: Control Objectives for Information and related Technology. Rolling Meadows: IT Governance Institute. (2007)

[Jak08] JAKOBSEN C. R., JOHNSON K. A.: "Mature Agile with a twist of CMMI", Proceedings of AGILE 2008 Conference (AGILE'08), pp. 212-217. (2008)

[Jor04] JØRGENSEN M.: »A review of studies on expert estimation of software development effort«. Journal of Systems and Software, 2004, Vol. 70, No.1-2, str 37–60. (2004) http://simula.no/research/se/publications/SE.4.Joergensen.2004.c/simula_pdf_file <na dan: 17.12.2012>

[Kan07] KANE B.: »Estimating and Tracking Agile Projects«. PM World Today, Vol. IX, Issue V (2007)

[Kta10] KTATA Q., LEVESQUE G.: "Designing and Implementing a Measurement Program for Scrum Teams: What do agile developers really need and want?", Proceedings of C3S2E-10, Montreal, Canada, 2010, str 101–1077. (2010)

[Kue00] KUENG P.: »Process performance measurement system: a tool to support process-based organizations«, Total Quality Management, Vol. 11, No. 1 (2000). http://www.peterk.ch/Kueng_2000.pdf <na dan: 17.12.2012>

[Lay12] LAYTON M.: »Agile Project Management for Dummies«. John Wiley & Sons, Inc. (2012).

[Lac10a] LACEY, Mitch: »Scrum for Managers«, predstavitev na Microsoft Corporation/TechTalk, Zurich Switzerland, March 2010 (2010). http://channel9.msdn.com/Events/TechEd/Europe/2010/ARC206 <na dan: 17.12.2012>

[Lev09] LEVINSON M.: »Recession Causes Rising IT Project Failure Rates« http://www.cio.com/article/495306/Recession_Causes_Rising_IT_Project_Failure_Rates_ <na dan: 17.12.2012>

[Mah05] MAHNIČ V., DRNOVŠČEK S.: »Agile software project management with scrum«. Eunis 2005 Conference, 20th-24th June, 2005, The University of Manchester : Manchester: The University of Manchester, IT Services Division, 2005, str. 1-6. (2005)

[Mah07a] MAHNIČ V., VRANA I.: »Using stakeholder driven process performance measurement for monitoring the performance of a Scrum based software development process«. Elektrotehniški vestnik, Ljubljana, vol. 74, no. 5, str. 241-247 (2007). http://ev.fe.uni-lj.si/5-2007/Mahnic.pdf <na dan: 17.12.2012>

[Mah07b] MAHNIČ V., ŽABKAR N.: »Introducing CMMI measurement and analysis practices into scrum-based software development process«. International journal of mathematics and computers in simulation, vol. 1, no. 1, str. 65-72 (2007). http://www.naun.org/multimedia/NAUN/mcs/mcs-11.pdf <na dan: 17.12.2012>

[Mah08a] MAHNIČ V., ŽABKAR N.: »Measurement repository for Scrum-based software development process«. V: GREBENNIKOV, Alexander (ur.), ZEMLIAK, Alexander (ur.). Modern topics of computer science : proceedings of the 2nd WSEAS International Conference on Computer Engineering and Applications (CEA'08), Acapulco, Mexico, January 25-27, 2008, (Electrical and computer engineering series). [S. l.]: WSEAS Press, cop. 2008, str. 23-28 (2008). http://www.wseas.us/e-library/conferences/2008/mexico/cea/1-%20CEA.pdf <na dan: 17.12.2012>

[Mah08c] MAHNIČ V., ŽABKAR N.: »Using COBIT indicators for measuring scrum-based software development«. WSEAS Trans. Comput., Oct. 2008, vol. 7, no. 10, str. 1605-1617 (2008). http://www.wseas.us/e-library/transactions/computers/2008/27-1239.pdf <na dan: 17.12.2012>

[Mah09] MAHNIČ V. GEORGIEV S., JARC T.: »Poučevanje metode Scrum v sodelovanju s podjetjem za razvoj programske opreme = Teaching the Scrum method in cooperation with a software development company«. V: RAJKOVIČ, Vladislav (ur.), BERNIK, Mojca (ur.), DINEVSKI, Dejan (ur.), URBANČIČ, Tanja (ur.). Vzgoja in izobraževanje v informacijski družbi :

zbornik konference : conference proceedings. Ljubljana: Ministrstvo za šolstvo in šport: Institut Jožef Stefan: Zavod Republike Slovenije za šolstvo; Kranj: Fakulteta za organizacijske vede, 2009, str. 243-255 (2009).

[Mah10a] MAHNIČ V., GEORGIEV S., JARC T.: »Teaching Scrum in cooperation with a software development company«. Organizacija (Kranj), 2010, vol. 43, no. 1, str. 40-48 (2010). http://organizacija.fov.uni-mb.si/index.php/organizacija/article/view/328/656 <na dan: 17.12.2012>

[Mah10b] MAHNIČ V.: »Teaching Scrum through team-project work: students' perceptions and teacher's observations«. Int. j. eng. educ., 2010, vol. 26, no. 1, str. 96-110 (2010).

[Mah11a]: MAHNIČ V.: " Problemi in reštive pri uvajanju metode Scrum v process razvoja programske opreme", Zbornik konference "Dnevi slovenske informatike", Portorož, Slovenia (2011)

[Mah11b]: MAHNIČ V.: »A Case Study on Agile Estimating and Planning using Scrum«, Electronics and electrical engineering, Vol 111, No 5 (2011), Kaunas, Litva. (2011) http://www.ee.ktu.lt/journal/2011/05/29__ISSN_1392-1215_A%20Case%20Study%20on%20Agile%20Estimating%20and%20Planning%20using%20Scrum.pdf <na dan: 17.12.2012>

[Mah12a]: MAHNIČ V.: »A Capstone Course on Agile Software Development Using Scrum«. IEEE Transactions on Education, Feb. 2012, Volume: 55, Issue: 1, str. 99-106. (2012)

[Mah12b]: MAHNIČ V., HOVELJA T.: »On using planning poker for estimating user stories«. The Journal of Systems and Software, Vol. 85, no. 9 (Sep. 2012), str. 2086-2095. (2012)

[Mah12c]: MAHNIČ V., ŽABKAR N.: »Measuring Progress of Scrum-based Software Projects«. Electronics and electrical engineering, Vol 18, No 8 (2012), Kaunas, Litva, str. 73-76. (2012)

[Mah12d]: MAHNIČ V.: »Introducing Scrum into the Development of a News Portal". Proceedings of Advances in applied information science, 12th WSEAS International Conference on Applied Informatics and Communications (AIC '12), Istanbul, Turkey, August 21-23, 2012. Str. 109-114. (2012) http://www.wseas.us/e-library/conferences/2012/Istanbul/AICBE/AICBE-15.pdf <na dan: 17.12.2012>

[Man05] MANN C., MAURER F.»A Case Study on the Impact of Scrum on Overtime and Customer Satisfaction«. Proceedings of the Agile Development Conference (ADC'05). IEEE Computer Society Washington, DC, ZDA. Str. 70-79. (2005) http://ase.cpsc.ucalgary.ca/uploads/Publications/MannMaurerAU2005.pdf <na dan: 17.12.2012>

[May09b] MAYER T.: »Scrum Roles – an abstraction", October 8, 2009 (2009). http://agileanarchy.wordpress.com/2009/10/08/scrum-roles-an-abstraction/ <na dan: 17.12.2012>

[McB00] McBREEN P.: »Questioning Extreme Programming«. Addison-Wesley, ZDA (2000).

[Mel05] MELNIK G., MAURER F.: »A Cross–Program Investigation of Students' Perceptions of Agile Methods«. Proceedings of the 27th International Conference on Software Engineering. St. Louis, ZDA, 2005. Str. 481–487 (2005)

[Mil11] GLAVAN MILANOVIĆ Lj.: »Understanding Process Performance Measurement Systems«. Business Systems Research. 2011, Vol.2, No.2, 1-56. Str 25-38. (2011)

[Mir10] MIRANDA E., BOURQUE P.: »Agile monitoring using the line of balance«. Journal of Systems and Software archive. Volume 83 Issue 7, July, 2010. Elsevier Science Inc. New York, NY, ZDA. Str. 1205-1215 (2010) http://mse.isri.cmu.edu/software-engineering/documents/faculty-publications/miranda/pdf5.pdf <na dan: 17.12.2012>

[Moe09] MOE N. B., DINGSØYR T., DYBÅ T.: »Overcoming Barriers to Self–Management in Software Teams«. IEEE Software, 2009. Vol. 26. No. 6. Str. 20–26 (2009) http://www.unix.eng.ua.edu/~kw5639/PDFs/selfmanagement.pdf <na dan: 17.12.2012>

[Mol08] MOLØKKEN–OSTVOLD K., HAUGEN N. C., BENESTAD H. C.: »Using planning poker for combining expert estimates in software projects«. Journal of Systems and Software, 2008.Vol. 81. No. 12. Str. 2106–2117 (2008)

[Mou10] MountainGoat: »A Reusable Scrum PresentationScrum presentation«. http://www.mountaingoatsoftware.com/scrum/a-reusable-scrum-presentation (v slovenščini) <na dan: 17.12.2012>

[NAO12] The National Audit Office: »A snapshot of the use of Agile delivery in central government«, 2012. http://www.nao.org.uk/publications/1213/use_of_agile_delivery.aspx <na dan: 17.12.2012>

[Nic09] NICOLETTE D.: »Agile Metrics«, predstavljeno na konferenci Agile 2009 (2009). http://davenicolette.wikispaces.com/Agile+Metrics <na dan: 17.12.2012>

[Piv96] PIVKA M.: »Kakovost v programskem inženirstvu«. DESK, Izola (1996).

[PMI08a] PMI: »A Guide to the Project Management Body of Knowledge (PMBOK Guide)«. 4. izdaja. Newtown Square: Project Management Institute. (2008)

[PMI08b] PMI: »Vodnik po znanju projektnega vodenja (PMBOK vodnik): tretja izdaja«. Moderna organizacije v okviru Univerze v Mariboru, Fakulteta za organizacijske vede, Kranj, 393 str. (2008)

[Pre12] PREIS A.: »Integration Evaluation of Scrum and CMMI - Theoretical Survey at the Project-Management Level«. Scientific Report Organizational Science, Vienna, Austria. (2012) http://www.arminpreis.at/files/2012/pub/PREIS-Integrate-CMMI-Scrum-Projectmanagement.pdf <na dan: 17.12.2012>

[Pro10a] PROULX, Martin: »Forrester Reports "Agile Development: Mainstream Adoption Has Changed Agility", February 8th, 2010 (2010). http://analytical-mind.com/2010/02/08/forrester-reports-agile-development-mainstream-adoption-has-changed-agility/ <na dan: 17.12.2012>

[Pro10b] PROULX, Martin: »Gartner Predicts 2010: Agile and Cloud Impact

Application Development Directions«, March 9th, 2010 (2010). http://analytical-mind.com/2010/03/09/gartner-predicts-2010-agile-and-cloud-impact-application-development-directions/ <na dan: 17.12.2012>

[Pro10c] PROULX M.: »Yet Another Agile Maturity Model (AMM) – The 5 Levels of Maturity«, July 12th, 2010 (2010). http://analytical-mind.com/2010/07/12/yet-another-agile-maturity-model-the-5-levels-of-maturity/ <na dan: 17.12.2012>

[Qas12] QASAIMEH M.: »Auditing for ISO 9001 requirements in the context of agile software processes«, doktorska disertacija, ÉCOLE DE TECHNOLOGIE SUPÉRIEURE UNIVERSITÉ DU QUÉBEC. (2012) http://espace.etsmtl.ca/1056/1/QASAIMEH_Malik.pdf <na dan: 17.12.2012>

[Raw06] RAWSTHORNE D.: »Calculating Earned Business Value for an Agile Project« http://danube.com/system/files/CollabNet_WP_Earned_Business_Value_041910.pdf <na dan: 17.12.2012>

[Raw10] RAWSTHORNE D.: »Monitoring Scrum Projects with AgileEVM and Earned Business Value (EBV) Metrics«. http://danube.com/system/files/CollabNet_WP_AgileEVM_and_Earned_Business_Value_Metrics_032510.pdf <na dan: 17.12.2012>

[Rob08] ROBILLARD, P. N., DULIPOVICI, M.: »Teaching Agile versus Disciplined Processes«. International Journal of Engineering Education, vol. 24, nr. 4, str. 671-680 (2008).

[Rož03] ROŽANC, Igor: »Model za kvantitativno spremljanje postopka razvoja programske opreme v skladu z zahtevami četrte stopnje modela zrelostnih stopenj programske opreme«. Doktorska disertacija, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani. (2003) http://eprints.fri.uni-lj.si/783/1/Rozanc-naslov.pdf <na dan: 17.12.2012>

[Run03] RUNESON P.: »Using students as experiment subjects – an analysis on graduate and freshmen student data«. Proceedings of the 7th International Conference on Empirical Assessment in Software Engineering. Keele University, UK. Str. 95–102. (2003)

[Rus09] RUSK J.: »Earned Value for Agile Development«, DoD Software Tech News, Vol. 12, No. 1. (2009) http://www.agilekiwi.com/EarnedValueForAgileProjects.pdf <na dan: 17.12.2012>

[Sch02] SCHWABER K., BEEDLE M.: »Agile Software Development with Scrum«. Prentice Hall , ZDA. (2002)

[Sch04] SCHWABER, Ken: »Agile Project Management with Scrum«. Microsoft Press, ZDA. (2004)

[Sch05] SCHATZ B., ABDELSHAFI I.: »Primavera Gets Agile: A Successful Transition to Agile Development«, IEEE Software - SOFTWARE , vol. 22, no. 3, pp. 36-42. (2005)

[Sch10c] SCHWABER, Ken: »Interview« (Dan Mezick). Infoq, avgust 2010, september 2010 (2010) http://www.infoq.com/news/2010/08/kenschwaber-interview-part1 <na dan:

17.12.2012>
http://www.infoq.com/news/2010/09/kenschwaber-interview-part2 <na dan: 17.12.2012>

[Sch11] SCHWABER K., SUTHERLAND J.: »The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game«, (1999-2011), zadnjič spremenjeno oktobra 2011 (2011) *(The Scrum Guide is the official Scrum Body of Knowledge.)* http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf <na dan: 17.12.2012>

[Scr12] ScrumAlliance: »What is Scrum?« (2012) http://scrumalliance.org/pages/what_is_scrum <na dan: 17.12.2012>

[Shu02] SHUKLA, A., WILLIAMS, L.: »Adapting Extreme Programming For A Core Software Engineering Course«. Proceedings of the 15th Conference on Software Engineering Education and Training (CSEET'02), str. 184-191 (2002) http://collaboration.csc.ncsu.edu/laurie/Papers/XPCORE.PDF <na dan: 17.12.2012>

[Sid07] SidkyA.: "A Structured Approach to Adopting Agile Practices: The Agile Adoption Framework," Doctoral Dissertation, Computer Science, Virginia Tech, Blacksburg, 2007. http://scholar.lib.vt.edu/theses/available/etd-05252007-110748/unrestricted/asidky_Dissertation.pdf <na dan: 17.12.2012>

[Ste03] STEPHENS M., ROSENBERG D.: »Extreme Programming Refactored: The Case Against XP«. Apress, ZDA (2003).

[Str10] STRANGE M.: "Scrum: WWDD? (What Would Deming Do?)", presentation at the Orlando Scrum Gathering, March 9, 2010 (2010) www.scrumalliance.org/resource_download/1472 <na dan: 17.12.2012>

[Sul06] SULAIMAN T., BARTON B., BLACKBURN T.: »AgileEVM - Earned Value Management in Scrum Projects«. Proceedings of AGILE 2006 Conference (AGILE'06). Str. 10-16. (2006) http://agileadvantage.com/wp-content/uploads/2011/11/Sulaiman-Barton-Blackburn-AgileEVM.pdf <na dan: 17.12.2012>

[Sut04] SUTHERLAND J.: "Agile Development: Lessons learned from the first Scrum". (2004) http://www.scrumalliance.org/resources/35 <na dan: 17.12.2012>

[Sut07] [Sut07] SUTHERLAND, J. et al.: "Scrum and CMMI Level 5: The Magic Potion for Code Warriors", Proceedings of AGILE 2007, pp. 272-278 (2007) http://www.systematic.com/Files/IS%20files/Downloads/Articles/Articles%20in%20English/Scrum%20and%20CMMI%20Level%205%20-%20The%20Magic%20Potion%20for%20Code%20Warriors.pdf <na dan: 17.12.2012> http://agile2007.agilealliance.org/downloads/presentations/sutherexp_425.pdf <na dan: 17.12.2012>

[Sut09] SUTHERLAND J.: "Nokia test, aka »The ScrumButt test«« (2009). http://jeffsutherland.com/nokiatest.pdf <na dan: 17.12.2012> http://scrum.jeffsutherland.com/2009/07/nokia-test-online.html <na dan:

17.12.2012>

[Sut09a] SUTHERLAND J., DOWNEY S., GRANVIK B.: "Shock Therapy: A Bootstrap for Hyper-Productive Scrum" (2009) http://jeffsutherland.com/SutherlandShockTherapyAgile2009.pdf <na dan: 17.12.2012>

[Sut09b] SUTHERLAND J.: "Practical Road Map to Great Scrum: Systematically Achieving Hyperproductivity (2009) http://jeffsutherland.com/PracticalRoadmapGTFS25Nov2009.pdf <na dan: 17.12.2012>

[Sza04] SZALVAY V.: "An Introduction to Agile Software Development«, White paper, Danube/Collabnet (2004) http://www.danube.com/docs/Intro_to_Agile.pdf <na dan: 17.12.2012>

[Tak86] TAKEUCHI H., NONAKA I.: "The New New Product Development Game" »Agile Development Is Hitting the Mainstream, Says Forrester«, Harvard Business Review, January-February 1986 (1986)

[Ure12]: UREVC J., ŠTEBE R., MAHNIČ V.: "Izkušnje z uvajanjem metode Scrum v časopisni hiši »Delo««. DSI 2012, Portorož. (2012) http://www.dsi2012.si/upload/predstavitve/140_Vodenje%20projektov%20in%20upravljanje%20odnosov%20z%20izvajalci/46_00_Izku_nje%20z%20uvajanjem%20metode%20Scrum%20v%20_asopisni%20hi_i%20Delo%20.pdf <na dan: 17.12.2012>

[Val10] VALDEZ M. T., AGREIRA C. F., FERREIRA C. M., MACIEL BARBOSA F.: »Teaching, Learning and Exploring the Use of Project–Based Learning«. Electronics and Electrical Engineering. Kaunas: Technologija, 2010. No. 9(105). Str. 117–120 (2010) http://www.ee.ktu.lt/journal/2010/9/27__ISSN_1392-1215_Teaching%20Learning%20and%20Exploring%20the%20Use%20of%20Project-Based%20Learning.pdf <na dan: 17.12.2012>

[Ver11] VersionOne: "State of Agile Survey, the state of Agile Development, 2011«, (2009) http://www.versionone.com/pdf/2011_State_of_Agile_Development_Survey_Results.pdf <na dan: 17.12.2012>

[XU06] XU S., RAJLICH V.: »Empirical Validation of Test-Driven Pair Programming in Game Development«. Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse (ICIS-COMSAR'06) (2006).

[Yap06] YAP M.: »Value based Extreme Programming«. Proceedings of AGILE 2006 Conference (AGILE'06) (2006).

[Wat07] WATSON, Hugh J., WIXOM, Barbara H.: »The Current State of Business Intelligence«. IEEE Computer, september 2007 (2007). http://eng.ui.ac.ir/~m.rezaei/techEnglish/WatsonWixom.pdf <na dan: 17.12.2012>

[Wik12a] [Wikipedia: »Agile software development« http://en.wikipedia.org/wiki/Agile_software_development <na dan: 17.12.2012>

[Wik12b] Wikipedia: »Lean software development« http://en.wikipedia.org/wiki/Lean_software_development <na dan: 17.12.2012>

[Wik12c] Wikipedia: »Scrum (development)« http://en.wikipedia.org/wiki/Scrum_(development) <na dan: 17.12.2012>

[Wu11] WU S.: »Running head: Traditional and Agile EVM Process«. Boston University School of Management. http://www.pmroi.com/storage/1_WuS_TEVMAEVMProcess_20110620_pmroi_web.pdf <na dan: 17.12.2012>

[Wys09] WYSOCKI R.: Effective Project Management (Traditional, Agile, Extreme). 5. izdaja. Indianapolis: Wiley Publishing, 734 str. (2009)

[Žab12] ŽABKAR N.: Notranje revidiranje uspešnosti in učinkovitosti projekta informacijske tehnologije. Zbornik referatov, 15. letna konferenca notranjih revizorjev, Podčetrtek, 15. in 16. november 2012. Ljubljana, Slovenski inštitut za revizijo, 2012. strani 107-124. (2012)